IBM Cognos Software Development Kit
Version 11.0.0

Custom Authentication Provider Developer Guide

Note

Before using this information and the product it supports, read the information in “Notices” on page 31.

Product Information

This document applies to IBM Cognos Software Development Kit Version 11.0.0 and may also apply to subsequent releases.

Contents

Introduction

This document is intended for use with the IBM® Cognos® Software Development Kit, which allows you to manage IBM Cognos processes and implement custom reporting solutions by using the cross-platform Web services, libraries, and programming interfaces provided with the IBM Cognos Software Development Kit. As a developer, you can use the Software Development Kit to create and modify reports and queries, schedule and deploy reports and other objects, and administer IBM Cognos Analytics servers and security.

This document provides background information to help you use the Custom Authentication Provider API to create a custom authentication provider, including a trusted sign-on provider. The trusted sign-on provider determines the user identity before connecting with an IBM Cognos authentication namespace.

Audience

The IBM Cognos Custom Authentication Provider is a Java™ application programming interface. To implement a custom authentication provider, you must have knowledge of the Java™ programming language.

We also recommend that you be familiar with the following:

- XPath, the XML search language of the World-Wide Web Consortium (W3C)
- The IBM Cognos Software Development Kit, particularly the BiBusHeader class and user authentication methods

Finding information

To find product documentation on the web, including all translated documentation, access IBM Knowledge Center (http://www.ibm.com/support/knowledgecenter).

Forward-looking statements

This documentation describes the current functionality of the product. References to items that are not currently available may be included. No implication of any future availability should be inferred. Any such references are not a commitment, promise, or legal obligation to deliver any material, code, or functionality. The development, release, and timing of features or functionality remain at the sole discretion of IBM.

Samples disclaimer

The Sample Outdoors Company, Great Outdoors Company, GO Sales, any variation of the Sample Outdoors or Great Outdoors names, and Planning Sample depict fictitious business operations with sample data used to develop sample applications for IBM and IBM customers. These fictitious records include sample data for sales transactions, product distribution, finance, and human resources. Any resemblance to actual names, addresses, contact numbers, or transaction values is coincidental. Other sample files may contain fictional data manually or machine generated, factual data compiled from academic or public sources, or data used with permission of the copyright holder, for use as sample data to develop sample applications. Product names referenced may be the trademarks of their respective owners. Unauthorized duplication is prohibited.

Accessibility features

Consult the documentation for the tools that you use to develop applications to determine their accessibility level. These tools are not a part of this product.

IBM Cognos HTML documentation has accessibility features. PDF documents are supplemental and, as such, include no added accessibility features.

Chapter 1. What's New?

This section contains a list of new, changed, and deprecated features for this and previous releases of IBM Cognos Custom Authentication Provider. It will help you plan your upgrade and application deployment strategies and the training requirements for your users.

To review an up-to-date list of environments supported by IBM Cognos products, including operating systems, patches, browsers, Web servers, directory servers, database servers, and application servers, visit the IBM Software Product Compatibility Reports page (http://www.ibm.com/support/docview.wss?uid=swg27042164).

New features in version 10.2.2

This section describes new features in this release of IBM Cognos Custom Authentication Provider.

New IBoundingSetProvider interface

This new interface allows users to share content among multiple tenants. See “IBoundingSetProvider Interface ” on page 15

New features in version 10.2.1

This section describes new features in this release of IBM Cognos Custom Authentication Provider.

Session failover capability

If your IBM Cognos Analytics server switches from an active to a standby Content Manager, users will now remained logged in when using the authentication providers supported by Cognos Analytics. If you have a custom authentication provider, you will have to modify it so that users remain logged when switching to a standby Content Manager. The JDBCSample program has been updated to include a RestorableJDBCSample (see “Custom authentication provider sample programs” on page 5) and a new interface, IRestorableVisa,(see “IRestorableVisa Interface ” on page 14) has been added with methods to support session failover.

New features in version 10.2.0

This section describes new features in this release of IBM Cognos Custom Authentication Provider.

JDBCSample sample program updated

The JDBCSample sample program now works with IBM DB2® databases, as well as Microsoft SQLServer databases. The sample has also been updated to demonstrate multi-tenancy support.

Development of a single sign-on authentication provider

A topic describes how to modify the JDBCSample sample program to support single sign-on authentication. See Chapter 4, “Development of a single sign-on authentication provider,” on page 23

New ITenantProvider interface

This new interface allows users to obtain tenant identifier information on accounts within a multi-tenancy environment. See “ITenantProvider Interface ” on page 14

Chapter 2. Authentication Providers

Authentication providers give access to authentication sources. By configuring an authentication provider in IBM Cognos Analytics, you define an authentication namespace.

The authentication namespace configuration is part of the Content Manager installation. The namespace can be configured to use full authentication provider, or a trusted sign-on provider with conjunction with a full authentication provider.

IBM Cognos Analytics does not use authentication providers for authorization purposes. Policies and permissions are stored in Content Manager and are managed by IBM Cognos Analytics.

IBM Cognos software supports the following authentication providers:

- IBM Cognos Series 7 namespaces
- LDAP version 3 directory servers
- Microsoft Windows native security (NTLM)
- SAP BW
- Active Directory Server
- Computer Associates eTrust SiteMinder (with LDAP and/or NTLM user directories)

If your authentication provider does not appear on this list, or the basic functionality of any of the listed providers does not meet your requirements, you can use the Custom Authentication Provider API to implement a custom authentication provider Chapter 3, “Developing a Custom Authentication Provider,” on page 5.

For more information about the IBM Cognos security infrastructure, see the *IBM Cognos Analytics Administration and Security Guide*.

For information about the IBM Cognos object model, see the *IBM Cognos Software Development Kit Developer Guide*.

A Full Authentication Provider

A full authentication provider implements all the functionality required by IBM Cognos software to communicate with an authentication source.

This includes

- User authentication using external authentication sources
- Namespace searches

  The searches can retrieve namespace objects and their properties, as required by IBM Cognos software. The objects can be users, groups, and roles, which are then used for authorization purposes in the Cognos namespace.
- Trusted credentials management
- Authentication provider configuration

The following diagram shows the IBM Cognos security architecture when a full authentication provider is implemented. It shows an example where Content Manager and Authentication Services has a provider, such as LDAP, and an authentication source, such as a Microsoft Windows Domain. The other branch has a custom provider and authentication source.

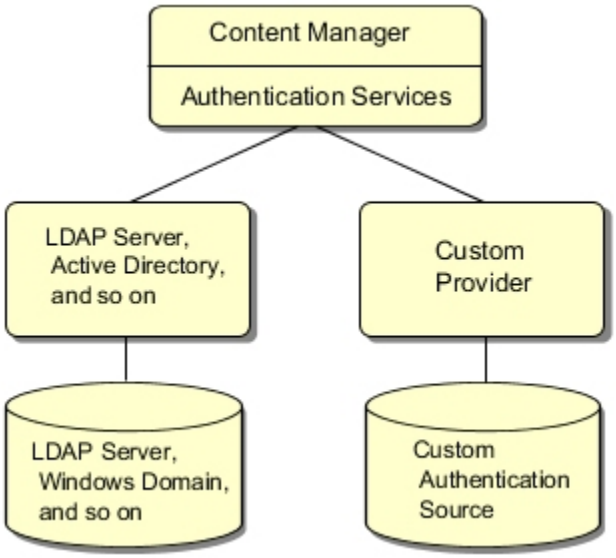

Figure 1. Security architecture for a full authentication provider

A Trusted Sign-on Provider

A trusted sign-on provider is used in IBM Cognos software to identify a user based on the session information from an authentication mechanism. After the user is identified, a full authentication provider is called to perform authorization.

A namespace for a trusted sign-on provider can be configured so that it is not selectable for authentication, which prevents it from being presented to users in lists of available namespaces. For information about configuring namespaces, see the *IBM Cognos Analytics Installation and Configuration Guide* or the *IBM Cognos Analytics Administration and Security Guide*.

The following diagram shows the IBM Cognos security architecture when a trusted sign-on provider is implemented.

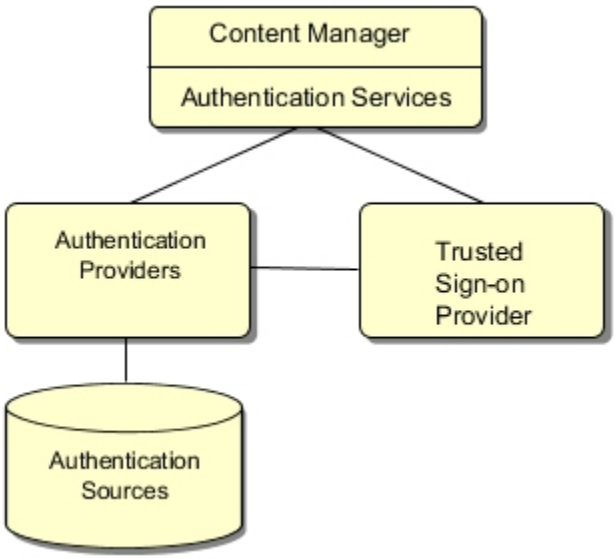

Figure 2. Security architecture for a trusted sign-on provider

Chapter 3. Developing a Custom Authentication Provider

You can develop your own authentication provider using the Custom Authentication Provider API to implement custom authentication methods, or to integrate with the methods not natively supported by your existing provider.

If you want your provider to support all the functionality required by IBM Cognos Analytics to communicate with an authentication source, create a full authentication provider "A Full Authentication Provider" on page 3. If you want your provider to perform user identification only, create a trusted sign-on provider "A Trusted Sign-on Provider" on page 4.

When you create a trusted sign-on provider, the following tasks are involved:

- Understanding the single sign-on functionality"Creating a Trusted sign-on Provider" on page 18
- Configuring the namespace interface "Configuring the Namespace Interface" on page 20
- Creating a manifest for the jar file "Creating a Manifest for the jar File" on page 21
- Registering an authentication listener "Register an Authentication Listener" on page 21

If you want, you can create a custom logon page to bypass the default IBM Cognos Analytics server logon page. For more information, see Appendix B, "Example: Creating a Custom Logon Page," on page 29.

For information about known issues associated with custom authentication providers, see Appendix A, "Troubleshooting," on page 27.

Prerequisites for developing a custom authentication provider

Before you begin, you must be familiar with the following concepts.

- Security concepts

  We assume that most application developers have a solid understanding of security concepts and that these concepts do not require explanation.

- IBM Cognos architecture

  You should be familiar with the IBM Cognos components, such as Content Manager and the security framework.

  For more information, see the *IBM Cognos Analytics Administration and Security Guide*, and the *IBM Cognos Software Development Kit Developer Guide*.

- Your current authentication mechanism and infrastructure

  You should be familiar with how your current authentication mechanism establishes trust by validating a user.

- The Java programming language

  You should be able to write Java code that can independently authenticate in your environment and respond to basic search queries.

Custom authentication provider sample programs

The following subdirectories are available in the *install_location*/sdk/java/AuthenticationProvider directory.

adapters
Contains sample implementation of the classes that you need to implement when writing a custom provider.

javadoc
Contains the generated API documentation in HTML.

JDBCSample
Contains a sample implementation of a custom authentication provider.

This directory also contains a version of this sample called RestorableJDBCSample that can handle restoring a session after Content Manager failover and demonstrates the changes necessary to implement failover in your custom provider.

This sample works with IBM Db2 and Microsoft SQL Server database applications.

lib
Contains the jar file for writing a custom provider (CAM_AAA_CustomIF.jar).

MultiTenancyTenantProviderSample
Contains a sample implementation of a multitenancy sign-on provider.

src
Contains the source code for the API. This may be useful because it provides context-sensitive help in many Java development environments, such as the Eclipse IDE.

TrustedsignonMappingSample
Contains a sample implementation of a mapping sign-on provider.

TrustedsignonReplaceSample
Contains a sample implementation of a trusted sign-on provider.

This sample expects a REMOTE_USER cookie and requests a cookie if one does not exist.

TrustedsignonSample
Contains a sample implementation of a trusted sign-on provider.

This sample expects a TRUSTED_sign-on_USER cookie and fails if one does not exist.

Running the Java samples on a Microsoft Windows operating system

Each sample subdirectory contains the following files:

- A build.bat file that builds the Java™ sample on Microsoft Windows operating systems.
- A readme.txt file that describes the sample and describes configuration steps that are required after compiling the sample program.
- One or more .java source files.

In addition, the AuthenticationProvider directory contains the following files:

- A build-samples.bat file that allows you to build all the samples at once on Microsoft Windows operating systems.

Procedure

1. Ensure that a Java Development Kit is installed.

   Note: The version of the JDK that you use must be one of the supported Java versions.
2. Add the location of the JDK bin folder to your path environment variable.
3. Run build-samples.bat to build all the samples or an individual build.bat to build a single sample.
4. Read the readme.txt file to get the instructions for configuring and using an individual sample.

Running the Java samples on UNIX or Linux operating systems

Each sample subdirectory contains the following files:

- A build.sh file that builds the Java™ sample on UNIX or Linux® operating systems.

- A readme.txt file that describes the sample and describes configuration steps that are required after compiling the sample program.
- One or more .java source files.

In addition, the AuthenticationProvider directory contains the following files:

- A build-samples.sh file that allows you to build all the samples at once on UNIX or Linux operating systems.

Procedure

1. Ensure that a Java Development Kit is installed.

   Note: The version of the JDK that you use must be one of the supported Java versions.
2. Add the location of the JDK bin folder to your path environment variable.
3. Run build-samples.sh to build all the samples or an individual build.sh to build a single sample.
4. Read the readme.txt file to get the instructions for configuring and using an individual sample.

Planning Your Implementation

Careful planning of development activities can greatly reduce the time and effort you spend developing a custom authentication provider.

The following checklist specifies all the activities involved.

__ • Create a flow diagram that represents a high level view of the authentication process and identifies the entry points to IBM Cognos Analytics.

Consider single sign-on and what session information is available in your IBM Cognos configuration.

For information about configuration, see the *IBM Cognos Analytics Installation and Configuration Guide*.

__ • Determine how to authenticate users "Defining User Authentication Methods" on page 8.

Decide how the authentication provider will identify individuals before they can sign-on to an application.

__ • Consider the objects you must use in your provider "Defining Namespace Searches" on page 15.

Each object must have a unique identifier. You map the objects you require in your provider to the objects exposed in IBM Cognos Analytics. The objects include groups, roles, folders, accounts, and the namespace folder.

For example, most security sources have fields for the following information about users:

- User name
- User identifier
- Password
- Phone number
- Email address
- Role membership
- Group membership

__ • Consider any objects, such as accounts or groups, and their properties, that you cannot store in your authentication source "Defining Namespace Searches" on page 15.

For example, some databases do not support fields for emails. If an empty value is retrieved, IBM Cognos Analytics assumes that this property is not available in the authentication source and saves it as its own property.

__ • Initialize your authentication source "Initializing the Authentication Source" on page 20.

Plan how your authentication provider will access the authentication source. Depending on the authentication source, you may have to initialize communication, and bind to the source by doing some configuration tasks.

__ • Create a manifest for the jar file “Creating a Manifest for the jar File” on page 21.

__ • Consider whether you want to support user session failover from an active to a standby Content Manager. If you do not support user session failover and the Content Manager switches to a standby Content Manager, users logged into all namespaces will be logged out, not just those logged into the custom authentication provider.

Defining User Authentication Methods

The session information for an authenticated user in IBM Cognos Analytics is maintained in a passport. The passport contains one or more visas. A visa exists for each namespace in which the user is currently authenticated. The passport remains active for the duration of a session and has a configurable idle time-out setting. IBM Cognos Analytics sets a session cookie in the user's browser that refers to the passport.

The process of authenticating a user in the IBM Cognos environment includes the following:

- Passport verification or creation

  A passport is required to process all requests in the IBM Cognos environment. The passport is created for a user after the first successful authentication.

  Named users must log on to one of the configured namespaces. The logon process depends on the authentication methods available for a namespace. For example, an LDAP namespace may be configured to prompt for a user ID and password, or to retrieve identity information for the user from an HTTP header variable.

- Creation of a passport visa

  After a successful logon, a visa is created to hold the namespace-specific user information for the duration of a session. Users can have one visa per session for each namespace to which they successfully log on.

Authentication Requests: Flow Scenarios

When you create a custom authentication provider, you must decide how your provider will process authentication (logon) requests.

This document provides flow scenarios that describe how different types of authentication requests are processed. Choose the scenario that applies to you, depending on your provider type and implementation.

In IBM Cognos Analytics, the authentication provider is called for logon requests if

- Anonymous access is disabled
- A Software Development Kit application invokes the LogonAs action

  For information about IBM Cognos Software Development Kit actions, see the *IBM Cognos Software Development Kit Developer Guide*.

When an authentication request comes from IBM Cognos Analytics, there are three interfaces through which it is delivered. It can be an interactive IBM Cognos Connection (browser) session, an IBM Cognos Mashup Service session, or a Cognos Software Development Kit application.

The followingidentification information objects may be used.

Form fields
HTML form elements used for data input, such as a user name and password, from a Web page or a Cognos Mashup Service session.

Credentials
Identification information, typically a user ID and password, provided by the Cognos Software Development Kit program logon method or by Content Manager to run scheduled reports and jobs.

Cookies
The cookie values store the information from the browser session.

Environment variables
The environment variable values store the information from the entry point gateway.

Trusted environment variables
The environment variable values signed by the entry point gateway.

Note: In this discussion, the term entry point is used to refer to an IBM Cognos gateway. The type of entry point depends on your IBM Cognos configuration and can be one of: CGI gateway, ISAPI gateway, Apache MOD gateway, Servlet gateway or Servlet dispatcher. For more information, see "Trusted sign-on Provider and Single sign-on" on page 19.

The following diagram shows the flow of an authentication request.

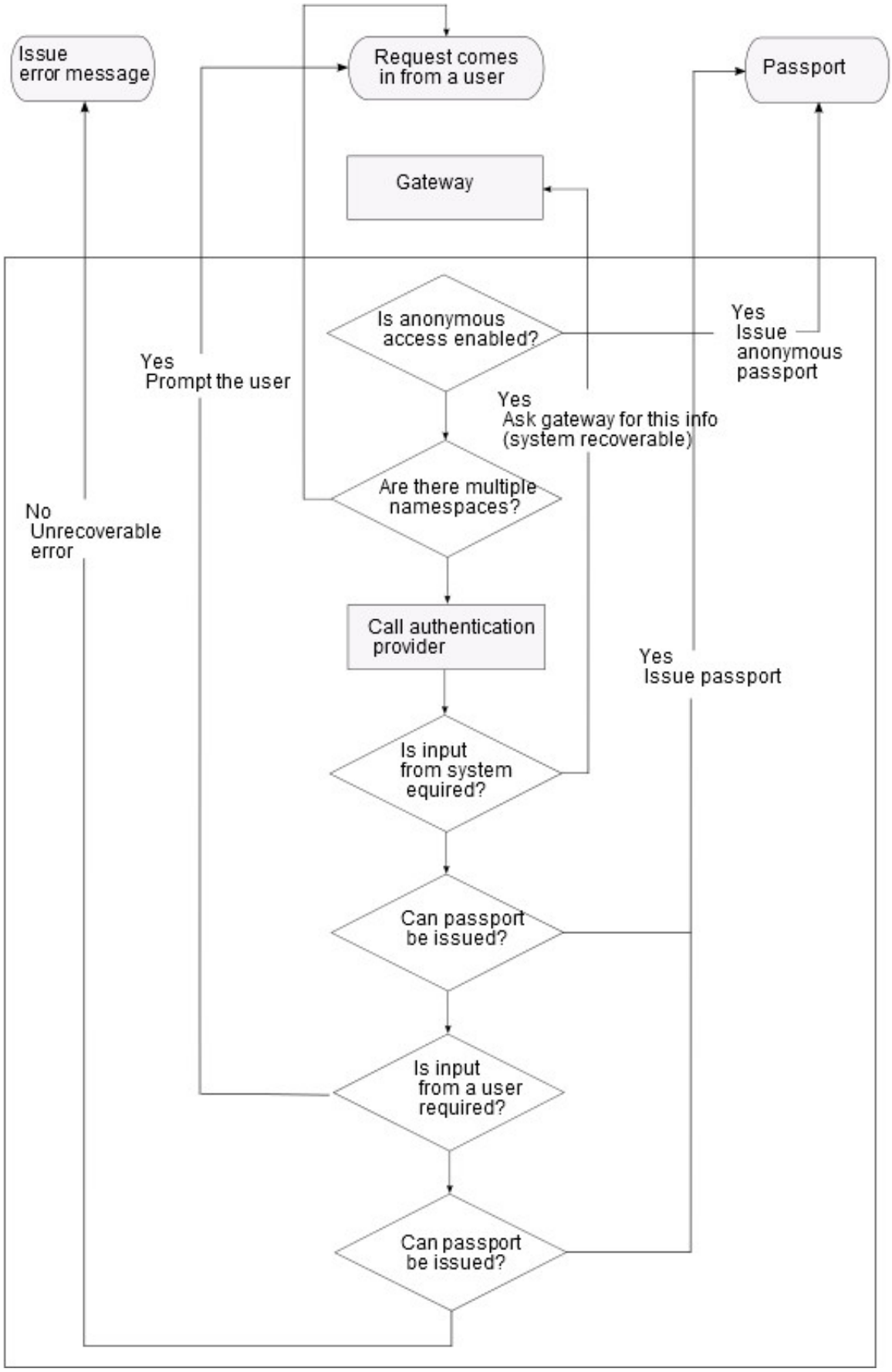

Figure 3. Authentication request flow

Scenario 1: Processing Requests That Contain All Required Logon Information

An authentication request is sent to the provider with enough information to authenticate the user in the source namespace. When the user is successfully authenticated, the provider creates a visa. The IBM Cognos Analytics server appends the visa to its passport and sets the passport ID cookie in the browser.

The following examples meet these criteria:

- The request includes information for single sign-on with cookies. For example, it includes a session cookie.
- A Software Development Kit program sends a request with the logon credentials in the Credential element of the BiBusHeader section.

The steps when the request contains all required information to authenticate a named user are shown here.

1. The gateway makes a request to the dispatcher for a resource.

   No passport is attached to the request.
2. The dispatcher makes a request to authentication services.
3. Authentication services calls the provider to authenticate the user.
4. The user is authenticated by the provider and a visa is created.
5. Authentication services issues a passport and adds the newly created visa to it.
6. A reference to the passport is attached to the request. This reference is stored in the browser as a session cookie.

Scenario 2: Processing Requests When Input From a User Is Required

If an authentication request does not contain all required information, a user must be prompted for the information.

The provider creates a user-recoverable exception, which includes a prompt for the user credentials. For example, it can be an HTML form with fields for a user name and password. Next, the IBM Cognos Analytics server prompts the user for the required information. After the information is entered, a response is returned. After the user-recoverable exception is successfully processed, the request flow continues as described in “Scenario 1: Processing Requests That Contain All Required Logon Information” on page 11

The steps when a request is made with no authentication information attached and no single sign-on in use are shown here.

1. The gateway makes a request to the dispatcher for a resource.

   No passport is attached to the request.
2. The dispatcher makes a request to authentication services.
3. Authentication services calls the provider, which generates the user-recoverable fault.
4. Authentication services forwards the fault back to the dispatcher to prompt for requested credentials.
5. The user is prompted and the data is sent back to authentication services.
6. Authentication services authenticates the authentication source with prompt values.
7. If the authentication process is successful, authentication services issues a passport and sets the session cookie in the browser.

Scenario 3: Processing Requests When Input From the System Is Required

The provider determines that additional information, which can be retrieved from the IBM Cognos gateway, is required.

Then, it creates a system-recoverable exception “Handling an Unrecoverable Error” on page 12 requesting the information, such as an environment variable.

The provider requests one or more environment variables from the gateway, and creates a message. The gateway or the dispatcher returns the data as a trusted environment variable, which ensures the source of the data. The message is required and is logged to the IBM Cognos logging function IPF. It must explain why the provider is throwing the exception.

After the system-recoverable exception is successfully processed, the request flow continues as described in "Scenario 1: Processing Requests That Contain All Required Logon Information" on page 11.

The steps for a request with no authentication information attached, but with single sign-on enabled are shown here.

1. The gateway makes a request to the dispatcher for a resource.

   No passport is attached to the request
2. The dispatcher makes a request to authentication services.
3. Authentication services calls the provider, which generates the system-recoverable fault.
4. Authentication services forwards the fault back to the dispatcher to prompt for the requested credentials.
5. The gateway collects the required information and the data is sent back to authentication services.
6. Authentication services authenticates the authentication source with prompt values.
7. Upon successful authentication, authentication services issues a passport and sets the session cookie in a browser.

Handling an Unrecoverable Error

If an unrecoverable error occurs, for example, when the authentication source is unavailable, the provider can throw an unrecoverable exception and terminate the logon action.

To handle an unrecoverable error, the recommended practice is to provide a message that, in its caption, informs that the error has occurred. Then, in the details portion of the message, you can add more specific information about the error. Because the error details are used by system administrators for diagnostic purposes, specifying a caption and a message is mandatory. You can use an HTML form to handle user-recoverable errors.

The possible user interface objects are described here.

Caption
Displays the text to the user, for example, to identify an error condition.

Message
Shows error details.

Authentication services logs the message data to the IBM Cognos Indication Processing Facility (IPF).

ReadOnlyDisplayObject
Shows the context information. For example, it can be a prompt that says: **Enter the password for smithj**

HiddenDisplayObject
Maintains context information across multiple requests.

TextDisplayObject
A regular input text box for a user name and other unsecured data.

SingleSelectDisplayObject
A drop-down list with one selection possible.

MultiSelectDisplayObject
A drop-down list with multiple selections possible.

VerifyTextNoEchoDisplayObject
An input text box used for a password change verification.

TextNoEchoDisplayObject
Echoes for entered text, such as *. Use for passwords and other secure data.

Implementing the User Authentication Interfaces

For a provider to successfully process user authentication requests, you must implement the following interfaces:

- INamespaceAuthenticationProvider2 interface
- The IVisa interface

You must also implement the required sub-interfaces.

For a complete reference of all available interfaces, see the javadoc help at *install_location*/sdk/java/AuthenticationProvider/javadoc.

INamespaceAuthenticationProvider2 Interface

The INamespaceAuthenticationProvider2 interface defines the interface implemented by an IBM Cognos custom authentication provider.

The following methods can be used in this interface.

init
Specifies that the authentication namespace is being placed into service.

This method is called only once for each instance of a namespace. The method must complete successfully before a namespace can receive any requests

destroy
Takes the namespace out of service.

This method is called after all the threads within the methods used by the namespace have exited, or after a time-out period has passed. After the engine calls this method, it does not call any methods for this namespace again. Any resources that are held, such as memory, file handles, or threads, must be cleaned up, and any persistent state is synchronized with the current state of the namespace in memory.

logon
Logs the user on to the authentication namespace.

logoff
Logs the user out of the authentication namespace.

search
Retrieves a subset of objects that exist in the specified authentication namespace.

In previous releases of IBM Cognos Analytics, the interface used to implement a custom authentication provider was named INamespaceAuthenticationProvider. The new interface, INamespaceAuthenticationProvider2, distinguishes between the information that comes from a credential and from a trusted credential. The old interface could not do that.

The INamespaceAuthenticationProvider interface is still supported for backward compatibility.

IVisa Interface

The IVisa interface is used to set and maintain the context for the user. After the Visa object is established, it is used for each new authentication provider request from the IBM Cognos Analytics server.

The IBM Cognos user can have only one Passport per session. In the Passport, there is a different Visa object for each namespace the user logs on to.

The following methods can be used in the IVisa interface.

generateCredential
Generates credentials for the user.

generateTrustedCredential
Generates trusted credentials for the user.

Use to maintain credential information used by IBM Cognos Analytics for scheduled jobs and reports.

getAccount
Returns account information for the logged user.

getGroups
Returns the groups of which the user is a member.

getRoles
Returns the roles of which the user is a member.

isValid
Verifies whether a Visa is still valid.

If the logon request does not contain sufficient information to validate the user, the generateCredential and generateTrustedCredential methods of the Visa may be returned with one of three types of exceptions:

- User-recoverable
- System-recoverable
- Unrecoverable error

IRestorableVisa Interface

The IRestorableVisa interface defines the interface for a version of an IVisa that can be restored when switching from an active to a standby Content Manager so that the users session is maintained without having to log in again.

The IRestorableVisa interface inherits the methods from the IVisa interface. The following additional methods can be used in the IRestorableVisa interface.

hasStateChanged
Determines whether the visa has changed. If the state has changed, the server will update any cached versions for failover

resetChangeIndicator
Resets the change indicator. A call to hasStateChanged should return false when called immediately after calling resetChangeIndicator.

The RestorableJDBCSample sample program contained in the JDBCSample sample directory uses the IRestorableVisa interface to illustrate the creation of a custom authentication provider that allows users to remain logged in when switching to a standby Content Manager. The comments in the RestorableJDBCSample.java and RestorableJDBCVisa.java files document the required steps. The following points are important and are illustrated in the RestorableJDBCVisa.java file.

- The custom authentication provider must monitor the sate of the visa and flag whenever it changes using the IRestorableVisa methods hasStateChanged and resetChangeIndicator.
- The java.io.Externalizable methods readExternal and writeExternal are used to externalize the user credentials and to submit them to the standby Content Manager.

ITenantProvider Interface

The ITenantProvider interface is used to implement a custom tenant identifier provider for use in a multitenant environment.

During system startup, the init(Map, String) method will be called once per configured instance. While the system is running, getTenantId(IAccount) will be called as accounts are logged on to. The destroy() method will be called at system shutdown.

By obtaining tenant identifiers for authenticated users, you could potentially associate a user session with a tenant.

The following methods are available in the ITenantProvider interface.

destroy
Destroys this ITenantProvider and frees resources held by it. This method is called during the system shutdown process.

getTenantId
Returns the tenant identifier that the provided account belongs to. An empty string will be returned for public accounts. A `tenantID` may contain any Unicode characters, with the exception of tabs, carriage returns, and line feeds. Leading and trailing spaces are trimmed, and consecutive internal spaces are reduced to one. A `tenantID` consisting only of spaces is reduced to an empty string, which is reserved for public content.

init
Initializes the ITenantProvider interface. This method is called during the system startup process.

If the initialization or `tenantID` retrieval process fails to complete, the `init` or `getTenantId` methods may be returned with an `UnrecoverableException` error.

IBoundingSetProvider Interface

The IBoundingSetProvider interface is used to share content among multiple tenants.

During system startup, the `init(Map, String)` method is called once per configured instance. While the system is running, `getBoundingSet(IAccount)` is called as accounts are logged on to. The `destroy()` method is called at system shutdown.

It is possible that multiple instances of the provider are created for a single namespace; this occurs when more than one CAM AAA service is running.

The following methods are available in the IBoundingSetProvider interface.

destroy
Destroys this IBoundingSetProvider and frees resources held by it. This method is called during the system shutdown process.

getBoundingSet
Returns the bounding set of tenant identifiers that the provided account belongs to. A null string is returned if the account has no bounding set. An empty string is returned for public accounts. A `tenantID` may contain any Unicode characters, with the exception of tabs, carriage returns, and line feeds. Leading and trailing spaces are trimmed, and consecutive internal spaces are reduced to one. A `tenantID` consisting only of spaces is reduced to an empty string, which is reserved for public content.

init
Initializes the IBoundingSetProvider interface. This method is called during the system startup process.

If the initialization or `tenantID` retrieval process fails to complete, the `init` or `getBoundingSet` methods may be returned with an `UnrecoverableException` error.

Defining Namespace Searches

IBM Cognos Analytics may request data that must be retrieved from the authentication source. The provider must respond to any search query that IBM Cognos Analytics sends.

The authentication namespace search is defined by a query. The query includes the following elements:

The following elements are used in an authentication namespace search query.

Search expression
Defines the objects that the search should return.

Properties
Defines object properties that should be returned with each returned object.

Sort Order
Specifies the sort order of the properties if they exist.

Locale information
Specifies the product and content locales, which define how the data must be displayed

Before you begin

To conduct the searches, IBM Cognos Analytics requires access to the following objects:

- Namespace
- User (account)
- Group
- Role
- Folder
- Object properties

Even if your authentication source does not have objects that can be mapped to those listed, the provider must respond to the queries. This may apply to a situation when, for example, a hierarchical concept of a folder does not exist in your authentication source because all objects are stored in a flat list.

Implementing the Namespace Search Interfaces

To successfully process search requests in your authentication namespace, you must implement the IQueryResult interface for the custom authentication provider.

The IQueryResult interface consists of the following interfaces:

- IQuery interface
- IQueryOption interface
- ISearchExpression interface
- ISearchStep interface

You must also implement the required sub-interfaces.

For a complete reference of all available interfaces, see the javadoc help at *install_location*/sdk/java/AuthenticationProvider/javadoc

IQueryResult Interface

The IQueryResult interface represents the result of a query.

The following tmethod is used by this interface.

getObjects
Returns all objects that are part of the query result.

IQuery Interface

The IQuery interface is used to build a provider query from an IBM Cognos query.

The following methods are used by this interface.

getContentLocale
Returns the preferred content locale for the session. The content locale determines the language and data format of the content returned from IBM Cognos Analytics.

getProductLocale
Returns the preferred product locale for the session. The product locale determines the language and data format for the IBM Cognos user interface.

getProperties
Returns the properties of the query object.

getQueryOption
Returns information about the query options to use, or null if there are no options specified.

getSearchExpression
Returns the search expression of the query object.

getSortProperties
Returns the sorting properties of the query, or null if the properties are not specified.

IQueryOption Interface

The IQueryOption interface defines options that modify the scope of a query.

The following methods are used by this interface.

getMaxCount
Specifies the number of objects requested by the search. The value -1 specifies that all objects will be returned.

getSkipCount
Specifies the start page or item for the search. The value 0 starts the count at the first item.

getRefProps
Returns the properties of the children of the searched object.

Use this method to request a list of children-objects. For example, for a user, this method returns the user name, account, and email, and for a group, a list of the group members that can include users and other groups.

ISearchExpression Interface

The ISearchExpression interface defines a search expression.

The following methods are used by this interface.

getObjectID
Returns the ID of the object on which the search expression is based, or null if the search is based on the root of the namespace.

getSteps
Returns all search filters that apply to the search expression, or null if there is no search filter.

ISearchStep Interface

The ISearchStep interface includes methods to get the axis and predicate for a search expression. The methods are used in exactly the same way as in XPath.

You can support multiple SearchStep interfaces in your provider. In combination, these equate to sub-searches. If you must search a multi-tiered security source, use multiple search steps to locate objects more than one level down in the hierarchy.

IBM Cognos Connection does not use more than one step. Authentication providers currently supported by IBM Cognos Analytics do not support more than one step, either. Only some Software Development Kit applications can generate queries containing more than one step, therefore a custom authentication provider is required to support such queries.

The following methods are used by this interface.

getAxis
Returns the location of an object in relation to its parent, child, ancestor, and descendant objects.

getPredicate
Returns a predicate expression used in a search.

Predicates are filters used to further locate an object beyond its axis location. Predicates can be

- Relational expressions, such as equals sign (=), greater than sign (>), less than sign (<).
- Function calls such as `contains`, `starts with`, and `ends with`.

- Conditional expressions
- Combinations of these expressions

For example, a predicate search for users whose first name is John, and whose email address contains GoSales.

Managing Trusted Credentials

A trusted credential contains authentication information that is stored in the Account object of the Custom Authentication Provider namespace in Content Manager for a later usage. The trusted credential is used to authenticate users during actions requiring authentication that are performed when the users are not logged on to IBM Cognos Analytics. For example, to run scheduled reports and jobs.

The TrustedCredential class contains name-value pairs. It is persisted beyond the current session.

To include the trusted credential functionality in your custom authentication provider, you must implement the ITrustedCredential interface "The ITrustedCredential Interface" on page 18.

For a complete reference of all available interfaces, see the javadoc help at *install_location*/sdk/java/AuthenticationProvider/javadoc.

The ITrustedCredential Interface

You must implement the ITrustedCredential interface for a provider to successfully process authentication requests for scheduled reports and jobs.

The following methods are used by this interface.

getCredentialNames
Returns all the credential names that exist, or null if the credentials do not exist.

getCredentialValue
Returns the value for the given credential name or null if it does not exist.

Creating a Trusted sign-on Provider

A trusted sign-on provider is used to determine the user's identity and communicate it to the appropriate authentication namespace configured using an authentication provider supported by IBM Cognos Analytics. The user's identity must be available to IBM Cognos Analytics before any secured object can be accessed in IBM Cognos Analytics.

You can create a trusted sign-on provider using this API to communicate between your custom authentication provider and your authentication source. The trusted sign-on provider can be implemented as an integral part of a full authentication provider "A Full Authentication Provider" on page 3 or independently from it.

With the Trusted sign-on Provider API, you can do the following:

- Add or remove an environment variable
- Add or remove a trusted environment variable for single sign-on
- Set a cookie
- Add or remove a credential
- Prompt a user for information (user recoverable exception)
- Prompt the system for information (system recoverable exception)
- Set a different namespace by setting an environment variable

The trusted sign-on provider receives an authentication request from IBM Cognos Analytics and modifies it, as required, to communicate with the provider authentication namespace. For example, it reads a cookie, decrypts it, parses out the user name, and sets the trusted environment variable, such as REMOTE_USER, to the required user name. Then, it sends the authentication request back to IBM

Cognos Analytics, specifying a full authentication namespace to process the request with the environment variable now set.

The authentication namespace determines the user membership information and user properties, performs namespace searches, and manages trusted credentials.

For a provider to successfully perform the trusted sign-on functionality, you must implement the INamespaceTrustedsignonProvider interface "The INamespaceTrustedsignonProvider Interface" on page 19.

Trusted sign-on Provider and Single sign-on

Depending on the IBM Cognos configuration, different session information is available for single sign-on functionality.

The session information available from the gateways that can be configured for IBM Cognos Analytics is described here.:

CGI gateway
CGI environment variables, HTTP header variables, REMOTE_USER

ISAPI gateway
HTTP header variables, REMOTE_USER

Apache MOD gateway
HTTP header variables, REMOTE_USER

Servlet gateway, or servlet dispatcher
HTTP header variables, REMOTE_USER, USER_PRINCIPAL (USER_PRINCIPAL is how the gateway or dispatcher exposes the J2EE user principal name to providers)

Authentication Credentials

Each supported authentication provider requires specific information to work with single sign-on.

The following table shows what type of information is required and where the information is stored for each currently supported authentication provider:

Table 1. Authentication provider credentials

Authentication provider	IBM Cognos gateway configuration	Single sign-on configuration
Series 7	Any IBM Cognos gateway	OS sign-ons with REMOTE_USER, OS sign-ons with the Trusted sign-on Plug-in
NTLM	Any IBM Cognos gateway	REMOTE_USER
LDAP V3	Any IBM Cognos gateway	External identity mapping using CGI environment variable, HTTP header variable, REMOTE_USER, and JAVA user principal.
SAP	Any IBM Cognos gateway	SAP session ticket
Active Directory	CGI or ISAPI gateway	REMOTE_USER
Custom providers	Any IBM Cognos gateway	Depends on your implementation

The INamespaceTrustedsignonProvider Interface

The INamespaceTrustedsignonProvider interface is the root interface used by the Trusted sign-on Provider.

You must also implement the appropriate sub-interfaces.

For a complete reference of all available interfaces, see the JavaDoc help at *install_location*/sdk/java/AuthenticationProvider/javadoc.

The following method is used by this interface.

processLogonRequest
Processes the authentication information used to identify a user.

Configuring the Namespace Interface

For IBM Cognos Analytics to successfully recognize a provider, you must implement the following interfaces for a full authentication provider and a trusted sign-on provider:

- INamespace interface "The INamespace Interface" on page 20
- INamespaceConfiguration interface "The INamespaceConfiguration Interface" on page 20

You must also implement the appropriate sub-interfaces.

For a complete reference about all available interfaces, see the javadoc help in the *install_location*/sdk/java/AuthenticationProvider/javadoc directory.

The INamespace Interface

The INamespace interface defines a namespace.

The following methods are used by this interface.

destroy
Takes the namespace out of service.

getCapabilities
Returns the capabilities of a namespace.

getNamespaceFormat
Returns the latest format supported by the namespace to uniquely identify a security object.

init
Specifies that an authentication namespace will be placed into service.

setNamespaceFormat
Sets the format that the namespace will use to uniquely identify a security object. For example, this method can be used to store version information for the namespace to manage future compatibility.

The INamespaceConfiguration Interface

The INamespaceConfiguration interface is used to query IBM Cognos Analytics for the configuration information about the authentication namespace.

The following methods are used by this interface.

getDisplayName
Returns the display name of the namespace.

getID
Returns the unique identifier for the namespace.

getInstallLocation
Returns the installation location of the current installation.

getServerLocale
Returns the server locale for the current installation.

Initializing the Authentication Source

When you have a working API, you can initialize your authentication source using IBM Cognos Configuration. You identify the custom authentication provider that is used to configure the authentication

namespace by selecting the Custom Java™ Provider, and specifying the name of the class that implements the provider.

If you use more than one Content Manager computer, you must configure identical authentication providers on each Content Manager computer. The type of authentication provider you select and the way you configure it must be identical on all computers for all platforms. The configuration must contain information that is accessible by all Content Manager computers.

Tip: The sample JDBCSample, located in the *install_location*/sdk/java/AuthenticationProvider/JDBCSample directory, shows the Microsoft SQL Server connection defined in the sample provider. In this case, the connection points to a configuration file. A sample properties file in the configuration folder demonstrates an IBM Db2 database connection, including comments that demonstrate a Microsoft SQL server configuration.

Deleting a Custom Authentication Provider Namespace

If the authentication provider namespace is no longer required, you can delete it using IBM Cognos Configuration.

When you delete a namespace, you can no longer log on to it. However, security data for the namespace remains in Content Manager until you permanently delete it in the portal. You can restore an inactive namespace if you need it again.

For more information, see the *IBM Cognos Analytics Installation and Configuration Guide.*

Creating a Manifest for the jar File

When you write a custom authentication provider, you must produce a jar file that contains the following information in its Manifest:

- Specification-Title: IBM Cognos Custom Provider SDK
- Specification-Version: 1.1
- Specification-Vendor: IBM Corp.

Register an Authentication Listener

The Custom Java authentication provider API includes the following authentication events:

- Logon

  This event is called after a successful logon. If a user logs on to multiple namespaces, there is a logon event for each namespace and the passportID remains the same.
- Logoff

  This event is called after a logoff request is issued from a user and before the passport is destroyed. A logoff cannot fail. There is one logoff event for each namespace to which a user is authenticated.
- LogonExpired

  This event is called when a passport expires and before the passport is destroyed. A logonExpired cannot fail. There is one logonExpired event for each namespace to which a user is authenticated.

The interface used to implement an IBM Cognos authentication listener is named IAuthenticationListener. For more information about this interface, see the documentation in the *install_location*/sdk/java/AuthenticationProvider/javadoc directory.

You can register an authentication listener for all namespaces or for specific namespaces.

Procedure

1. Go to the *install_location*/configuration directory.
2. Open the file AAA.properties.sample using a text editor.

3. Edit the file by typing the appropriate parameters instead of the *your java class* placeholders.
4. Save the file as AAA.properties.

Chapter 4. Development of a single sign-on authentication provider

A single sign-on authentication provider supports the use of environment variables to authenticate a user. The modification of the JDBCSample sample program to support single sign-on authentication is described here.

When an authentication request is passed to the authentication provider, the `authenticate()` method of the provider is called. This method handles the authentication process using the following steps.

1. Identifying the type of authentication data
2. Running the authentication data through the authentication source
3. Issuing a visa
4. Adding the visa to a passport

Here we discuss the first step of identifying the type of authentication data as the rest of the process is specific to the supported authentication source.

A practice for handling the identification of authentication data is to use nested `if` statements that check sequentially for trusted credentials, credentials, forms fields, and finally for single sign-on authentication. Only if no explicit authentication data is provided does the code handle single sign-on authentication.

Authentication methods

Each type of authentication data has a matching method in the `IBiBusHeader2` class to be used to retrieve it from the request which is passed to the `logon()` function. These methods can be found in the JDBCSample.java sample program and are shown here.

trusted credentials authentication type
`getTrustedCredentialValues()`

credentials authentication type
`getCredentialValues()`

forms fields authentication type
`getFormFieldValues()`

View the JDBCSample.java sample program to see how these methods work.

Environment variables

Environment variables can be read with either of two methods.

The `getEnvVarValue()` method reads the value of an HTTP header from the request. You can obtain a list of retrievable environment variables by turning on logging and capturing the SOAP request. This method does not trigger the authentication process and simply takes the value passed in the request. While this procedure can be used in non-CGI environments it is not secure. It is easy to inject HTTP headers into a request and a malicious user could add a `REMOTE_USER` header with any value to the request.

The `GetTrustedEnvVarValue()` method can be used to trigger the authentication process for the parameters passed to this method. This method requires that the HTTP header in the request sent to the provider be populated with values stored locally as environment variables. The authentication process reads the variable values from the environment at the entry point and returns it to the provider. This variable is retrieved from an IBM Cognos component and the provider trusts the value because it is transmitted signed and encrypted.

Note: You must throw the SystemRecoverableException and then the next time you are called, you call `getTrustedEnvVarValue()`.

Modification of the JDBCSample sample program

We can now describe how to modify the JDBCSample sample program to support single sign-on authentication.

We use the `getTrustedEnvVarValue()` method to read the environment variables. To start the authentication process an exception has to be returned to the entry point. This process must be explicitly handled in the provider code. The `getTrustedEnvVarValue()` method returns either the value of the variable as a string, or `NULL` if no value could be found. If `NULL` is returned, then there is no trusted value retrieved and the provider code raises a `SystemRecoverable` exception and requests the variable. From the point of the provider this request is now complete. When a new request arrives and the `GetTrustedEnvVarValue()` method is called again, it returns a string value. If this string value is empty, the single sign-on authentication failed and the provider falls back to form-based authentication by throwing a `UserRecoverable` exception that requests the user enter credentials, or exits with an `Unrecoverable` exception.

It is the provider that throws the `SystemRecoverable` exception and as part of that provides the name of the requested variable. The rest of the functionality, including encrypting the values and signing the proprietary headers used to transfer the data back and forth between the provider and the entry point, is shielded from the developer.

The JDBCSample sample program uses two `if` statements in sequence to check for the different types of logon data. Only if either the username or password is populated does the code proceed to verify the logon data or issue a visa. If neither the username nor the password have assigned values retrieved from logon data, the code throws a `UserRecoverable` exception that leads to the display of a logon screen if a web browser is the client.

We will add support for the fourth type of logon data, the trusted environment variable, in this example the `REMOTE_USER` variable.

The simple approach is to add a third `if` block after looking for credentials from the form fields. However, that implies that whatever happens in the form field `if` block has to populate username and password to pass the final test regardless of whether something has been retrieved in the last `if` statement. Single sign-on authentication typically does not provide passwords because the purpose of single sign-on authentication is to avoid having to provide the password multiple times, and it is a security risk to transmit passwords in clear text. We can assume that our single sign-on authentication does not provide a password. That implies that we must change the final `if` statement to allow it to pass only the user name in the case of single sign-on authentication.

However, if single sign-on authentication is supported, it can be used to generate trusted credentials as well. If a user is authenticated to an IBM Cognos Analytics server by single sign-on authentication, the logon data will only be a user name. If the provider is called to generate a trusted credential to store with a schedule, it can generate this credential based only on a user name. When that schedule is executed later, it is the trusted credentials, consisting only of a user name, that are presented to the provider. Once the provider supports single sign-on authentication it has to deal with trusted credentials that may contain only a user name, that is, even trusted credential-based authentication must be allowed with just a user name.

This procedure can be extended to the IBM Cognos Software Development Kit credential-based authentication as well. However, Cognos Software Development Kit clients usually have enough functionality to provide full credentials, which are user name and password.

We can change part of the code of the sample program and change the structure of the `if` statements from sequence to nested. This allows decisions on the contents of the logon data for each type individually. To pass the last `if` statement, we simply populate the password with the user name in case of single sign-on authentication.

The following code snippet shows a possible solution to this requirement. To see this code in context, view the following sample.

install_location`/sdk/java/AuthenticationProvider/JDBCSample/JDBCSample.java`

The two code sections added to support single sign-on authentication are delimited by comments.

```
  public IVisa logon(final IBiBusHeader2 theAuthRequest) throws
      UserRecoverableException, SystemRecoverableException,
      UnrecoverableException
  {
    JDBCVisa visa = null;

    // 1 - Look for trusted credentials
    Credential credential = JDBCSample.getTrustedCredentialValues(theAuthRequest);
    if (credential.isEmpty())
      credential = JDBCSample.getCredentialValues(theAuthRequest);

    if (credential.isEmpty())
      credential = JDBCSample.getFormFieldValues(theAuthRequest);


    //*** START NEW CODE FOR SINGLE sign-on ***

    if ( credential.isEmpty() && this.connectionManager.singlesign-onEnabled())
      credential = JDBCSample.getTrustedEnvironmentVariableValue( theAuthRequest );

    if ( credential.isEmpty() && this.connectionManager.singlesign-onEnabled())
    {
        // null implies the provider has to start the process so throw
           a SystemRecoverableException.
        // the SystemRecoverableException needs to have the name of the
           variable we look for in
        // the second parameter
        SystemRecoverableException e = new SystemRecoverableException(
                "Challenge for REMOTE_USER",
                "REMOTE_USER");
        throw e;
    }

    //*** END NEW CODE FOR SINGLE sign-on ***

    if (credential.isEmpty())
    {
      //Assume this is the initial logon and pass null for errorDetails
      generateAndThrowExceptionForLogonPrompt(null);
    }

    try
    {
      //
      // Create a Visa for the new user.
      //
      visa = new JDBCVisa();
      visa.init(this, this.connectionManager, credential.getUsername(),
         credential.getPassword());
    }
    catch (final UnrecoverableException ex)
    {
      final String errorDetails = JDBCSample.getErrorDetails(ex);

      // Something went wrong, probably because the user's credentials
      // are invalid.
      generateAndThrowExceptionForLogonPrompt(errorDetails);
    }
    return visa;
  }

    //*** START NEW CODE FOR SINGLE sign-on ***

protected static Credential getTrustedEnvironmentVariableValue
   (final IBiBusHeader2 authRequest)
  {
    final Credential credential = new Credential();
    final String[] userNames;
    userNames = authRequest.getTrustedEnvVarValue("REMOTE_USER");

    if ( userNames != null && userNames.length > 0 )
      credential.setUsername(userNames[0] );
// To simplify, we set the password to be the same as the username.
      credential.setPassword(userNames[0] );

    return credential;
  }
    //*** END NEW CODE FOR SINGLE sign-on ***
```

With this logic, the provider triggers the authentication process and retrieves the REMOTE_USER variable. If that variable has a value, it is transferred to username and password.

We then process the credentials and verify them with the authentication source.

This technique can be adopted to use any HTTP header variable. Cookies cannot be used because the entry point does not support them. Using cookies should be handled in a trusted service provider which then passes on to a full authentication provider which uses REMOTE_USER based single sign-on authentication, as demonstrated here.

The previous code snippet does not make the JDBCSample work. The sample code relies on passing a user name and password to the JDBC driver and only if those credentials are successfully used to establish a database connection is the visa issued. However if one creates a user that has the user name as a password this process will work.

Appendix A. Troubleshooting

This section describes known issues that you may encounter when working with the Custom Authentication Provider API, and includes suggestions for resolving them.

Problems with .jar Files

When working with a custom authentication provider, conflicts may occur between the open source software packages provided by IBM Cognos Analytics and the packages used by your organization. This applies to the Java™ archive (.jar) files in the *install_location*/webapps/p2pd/WEB-INF/lib directory, mainly commons-logging.jar and dom4j.jar. As a result, errors may occur when running reports.

Depending on your environment, perform one of the following sets of steps to resolve the problem.

Solution 1

1. Replace the commons-logging.jar and dom4j.jar files in the *install_location*/webapps/p2pd/WEB-INF/lib directory with the original copies of these files available on the IBM Cognos installation CD.
2. Restart the IBM Cognos service.

Solution 2

1. In the *install_location*/webapps/p2pd/WEB-INF directory, create a new directory named AAA/lib.
2. Copy all the .jar files used by your custom authentication provider, except for CAM_AAA_CustomIF.jar and CAM_AAA_CustomProxy.jar, into the *install_location*/webapps/p2pd/WEB-INF/AAA/lib directory.
3. Ensure that the *install_location*/webapps/p2pd/WEB-INF/lib directory contains only the original files provided by IBM Cognos Analytics. If needed, restore this directory to its original state using the IBM Cognos installation CD.
4. Restart the IBM Cognos service.

Appendix B. Example: Creating a Custom Logon Page

Administrators can create their own custom logon page to bypass the default logon page provided by the IBM Cognos Analytics server. This custom page is a form that accepts the user's logon parameters and passes them to the IBM Cognos server for authentication. The user is then redirected to IBM Cognos Connection.

You can create a new logon page, or leverage an existing page used by a different application.

Following is an example of a simple logon page. The required authentication variables and preferences are passed to IBM Cognos Connection using an HTML form.

```
<HTML>
<HEAD>
<TITLE></TITLE>
<SCRIPT>

function createCookie()
{
    var name = 'userId';
    var value
    value = document.forms(0).CAMUserId.value;
    document.cookie = name+"="+value+";path=/"
    + ";CRN=content"
    + ";Locale=en-us"
    + ";productLocale=en"
    + ";format=HTML"
    + ";timeZoneID=EST"
    + ";useAccessibilityFeatures=false"
    + ";skin=corporate"
    + ";listViewSeparator=none"
    + ";automaticPageRefresh=30"
    + ";showOptionSummary=true"
    + ";linesPerPage=15"
    + ";displayMode=list"
    + ";columnsPerPage=3"
    + ";showWelcomePage=true";
}
//createCookie();
</SCRIPT>

</HEAD>
<BODY>

<form action=http://server-name/ibmcognos/bi/v1/disp
    id=form1
    name=form1">

<input type=text name=CAMUserId value=""></input>
<input type=button value="Submit" id=submit1 name=submit1
    onclick="javascript:createCookie();submit();"></input>

</form>

</BODY>
</HTML>
```

Notices

This information was developed for products and services offered worldwide.

This material may be available from IBM in other languages. However, you may be required to own a copy of the product or product version in that language in order to access it.

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. This document may describe products, services, or features that are not included in the Program or license entitlement that you have purchased.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant 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 (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 may 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 Software Group
Attention: Licensing

3755 Riverside Dr.
Ottawa, ON
K1V 1B7
Canada

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 measurements 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.

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.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Depending upon the configurations deployed, this Software Offering may use session and persistent cookies that collect each user's

- name
- user name
- password

for purposes of

- session management
- authentication
- enhanced user usability
- single sign-on configuration
- usage tracking or functional purposes other than session management, authentication, enhanced user usability and single sign-on configuration

These cookies cannot be disabled.

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 https://www.ibm.com/privacy/us/en/.

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.

The following terms are trademarks or registered trademarks of other companies:

- Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.
- Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.
- 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.

Index

Special Characters

A

C

D

F

H

I

L

M

P

S

T