Implement seamless authentication between security systems and IBM Cognos BI, using these
guidelines. This document contains important information about the technical concepts and
backgrounds involved and the design of authentication and single sign-on (SSO) functionality
in IBM Cognos BI. It also discusses setting expectations regarding supported vs unsupported
environments and information that must be gathered for troubleshooting the IBM Cognos BI
side of things. This document is intended for security architects and administrators designing
authentication for a system includes Cognos BI.

Introduction

Purpose of Document
This document provides guidelines for implementing seamless authentication between security
systems and IBM Cognos BI. In particular this document contains important information about:

• The technical concepts and backgrounds involved
• The design of authentication and single sign-on (SSO) functionality in IBM Cognos BI
• Setting expectations regarding supported vs unsupported environments
• Information that must be gathered for troubleshooting the IBM Cognos BI side of things

Applicability
The target audience for this document are security architects and administrators designing
authentication for a system that IBM Cognos BI is a part of.

Exclusions and Exceptions
The document will not describe particular setups or steps to implement a certain integration other
than for the purpose of examples. For technical guidance or step-by-step instructions on how to
set up a particular integration, refer to the Security section in the Proven Practice area of the IBM
developerWorks Business Analytics zone (http://www.ibm.com/developerworks/analytics) as it is
possible that the integration you're looking for has been covered. If not, please contact the IBM
Business Analytics Proven Practice team or IBM Cognos Product Management.

Assumptions

It is assumed the reader is familiar with the contents of both the IBM Cognos BI Installation and
Configuration Guide and the IBM Cognos BI Administration and Security Guide. Both documents
are delivered with the product and are also available online in the IBM Cognos BI Information
Center. It is further assumed the reader has some familiarity with security concepts as they apply
to web applications.

Overview

IBM Cognos BI, being part of the Business Analytics portfolio of IBM, is more frequently being
integrated with other software solutions. One of the most common requirements is to provide a
seamless authentication experience to users, often referred to as SSO to Cognos.

This applies to, but is not limited to, IBM products such as IBM WebSphere Commerce, IBM Tivoli
WebSEAL, IBM WebSphere Portal, IBM Tivoli Common Reporting or IBM WebSphere in general.
In addition, there are integrations with third party authentication proxies such as eTrust, CA
SiteMinder and Oracle Oblix as well as web servers and/or application servers such as Microsoft's
IIS, Apache and SAP NetWeaver.

There are several challenges involved with providing SSO to IBM Cognos BI,

- Knowledge challenges
  - Authentication in IBM Cognos BI
  - IBM Cognos BI single sign-on concepts
- Technical challenges
  - Setup
  - Configuration
  - Testing
  - Troubleshooting
- State of support

The following sections of this document address these challenges by providing extensive insight
into the design concepts and techniques leveraged by IBM Cognos BI to system administrators
and security architects. The guidance provided enables administrators to deal with SSO
requirements and plot solutions for integrating authentication with IBM Cognos BI.

Authentication in IBM Cognos BI

This chapter is going to provide a comprehensive description of the concepts and components
involved with the authentication process in IBM Cognos BI.
Services, requests and sessions

IBM Cognos BI is a web application built on a service oriented architecture (SOA). All functionality is provided by a collection of independent, multi-instance web services communicating with clients and amongst each other using the SOAP protocol over a network. A special service, the Presentation Service, proxies requests for pure HTML based clients such as browsers.

A client, may it be a browser or some other client developed or based off the IBM Cognos BI Software Development Kit (SDK), sends requests to a valid IBM Cognos BI entry point using the HTTP protocol.

Usually this entry point is an IBM Cognos BI Gateway (GW) deployed to a web server but it could also be an IBM Cognos BI Dispatcher (DISP) Java servlet or a Servlet Gateway (SGW) hosted by an application server such as IBM WebSphere. Each request either explicitly or implicitly asks for a certain service to handle it and whenever there is no explicit target service being requested the Presentation Service is assumed. Each request sent to an IBM Cognos BI entry point will therefore have to be routed to an instance of the target service supporting this particular type of request.

Routing is handled by a Dispatcher, regardless of whether a request was sent to a Dispatcher directly or indirectly via a Gateway. Each Gateway will relay received requests to only one Dispatcher at a time.

The Dispatcher checks every incoming request for a sessionID which would indicate that this request is part of an already established HTTP session. If no sessionID is found, it assumes a new session is starting and creates a new sessionID. Next a requestID will be assigned to the request and both the sessionID and the requestID are added to the request. This enables identifying requests within a session and ensures unique requestIDs for each request as per security best practices.

With the services in a SOA being logically independent, authentication is handled at the HTTP session level rather than a per service level. Each service enforces the existence of an already authenticated session before accepting and handling a request in order to prevent unintended access. This guarantees the client sending the request is identified and linked to an identity.

As of IBM Cognos BI version 10, authentication functionality is centrally provided by the internal CAM_AsyncAAService which is only reachable indirectly through the Content Manager Service (CMS). Consequently all IBM Cognos BI services will employ the CMS to verify session authentication or to trigger authentication for unauthenticated sessions. The CMS is implemented as a Java servlet and is the service interface to the IBM Cognos Content Manager (CM), the central component with exclusive access to IBM Cognos BI's application repository known as the Content Store (CS).

As soon as a request is handled by a Dispatcher, it will first check the authentication status of the current session the received request belongs to. If a session doesn't exist yet, a new session ID will be implicitly created and the request automatically becomes the first request of that new session. If the session is not yet authenticated, the Dispatcher will employ the CAM_AsyncAAService to run the authentication process and pass the request off to it. The
CAM_AsyncAAAService service is a service interface to the Cognos Access Manager (CAM) sub-component which handles authentication and authorization (AA). Other CAM components that are not exposed through the CAM_AsyncAAAService service will handle tasks such as encryption (CAM-CRP). The AA sub-component of CAM will then employ authentication providers (AP), hosted in separate processes, to attach to external authentication sources to drive the process of session authentication. The details of this process will be covered later in the Authentication Process section but there are still some additional concepts that need to be introduced.

Illustration 1 depicts the schematic request flow and the services and components that are involved.

**Illustration 1: Schematics of components involved with IBM Cognos BI session authentication**

If the authentication succeeds, the session's authentication information is persisted in an internal Content Manager object and a base64-encoded, signed reference to this object is attached to the HTTP session.

All services can validate the authentication status of a session's request by calling the CAM_AsyncAAAService and providing the reference to the session's internal Content Manager object holding the authentication information. In the case where the session is found to be unauthenticated because the validation of the referenced object failed, the authentication process will be triggered again.

The session's authentication expires after a configurable period of idle time which can be specified in IBM Cognos Configuration. The configured timeout implies that the session's authentication object hasn't been refreshed by a validation request being sent to CAM_AsyncAAAService for that particular amount of time. Upon expiry the session's internal Content Manager object will be deleted thus invalidating the reference saved to the HTTP session.
Authentication Providers and Namespaces

IBM Cognos BI authentication works by deferring the actual authentication to external authentication sources through a piece of software known as an authentication provider.

Each authentication provider attaches to a specific type of authentication source such as a Lightweight Directory Access Protocol (LDAP) compliant server or a Microsoft Active Directory (AD) domain controller. Leveraging APIs provided by the authentication source, the authentication provider implements the necessary functionality to handle the authentication process and subsequently read security objects from the attached authentication source. It’s important to note that authentication providers only ever read from authentication sources but never write to them.

The security objects known to IBM Cognos BI are users, groups and roles. Authentication sources are expected at a minimum to support and store these three types of security objects. If an authentication source uses other or additional security objects to represent users, groups or roles, the authentication provider will have to map them to one the three types known to IBM Cognos BI.

If an authentication provider successfully authenticates to the authentication source based on credentials provided to it, this authentication suffices for IBM Cognos BI as well.

Many authentication providers will also provide support for single sign-on (SSO) by consuming some HTTP session-based token, thereby allowing seamless authentication between a non-IBM Cognos security layer and IBM Cognos BI.

Since authentication providers depend on the authentication source’s API to implement the functionality to attach to it and handle authentication, there may be platform related limitations regarding support for authentication providers. Not all authentication sources are available on all platforms supported by IBM Cognos BI, therefore their APIs are not supported on those platforms either. Refer to the Authentication Provider section of the Supported Software Environments page for the IBM Cognos BI product release in question (http://www.ibm.com/support/docview.wss?uid=swg27014782).

With the exception of one special type of provider to be explained in the next paragraph, each authentication provider gets configured by an instance of a Namespace configuration created in IBM Cognos Configuration. In the Namespace configuration, properties specific for that provider instance can be configured. The most important one being the NamespaceID, a string which uniquely identifies the authentication provider instance internally. On startup of the IBM Cognos BI CAM component, the configuration is scanned for Namespace configurations and the respective providers get initialized based on the specified settings. Once initialized, all authentication provider instances except for the one special type mentioned earlier expose the objects read from the external authentication source through an IBM Cognos BI Namespace. A Namespace is the representation of the external authentication source’s data mapped to the security objects relevant to IBM Cognos BI. For a single IBM Cognos BI system, multiple Namespaces can be configured at the same time, allowing access to IBM Cognos BI for users managed in heterogenous
authentication sources or even authenticated to more than one authentication source within a single session.

For every IBM Cognos BI system there is a single instance of a special, built-in Namespace called the **Cognos Namespace** which is initialized automatically and does not have a Namespace configuration object. It is not attached to an actual authentication source and does not contain any user objects - it only contains group and role objects. The Cognos Namespace is used to provide a logical mapping layer for security objects of Namespaces attached to external authentication sources and application defined authorization objects. For this purpose, one creates groups and roles in the Cognos Namespace and assigns members from all the other configured Namespaces.

The security best practice for IBM Cognos BI is to base off object security (authorization) on groups and roles from the Cognos Namespace only. This provides flexibility and portability since all authorization is only indirectly linked to external objects from the authentication sources.

As of IBM Cognos BI version 8, all out of the box authentication providers support the **Test** functionality which is available by right-clicking on a Namespace configuration element in IBM Cognos Configuration's explorer tree. This will initialize the provider using the configured settings but it may not exercise all the functionality the provider supports. It's meant to catch obvious configuration errors and connectivity issues only.

IBM Cognos BI out of the box delivers authentication providers for,

- LDAP v3 compliant servers
- Microsoft Active Directory
- IBM Cognos Series 7
- Microsoft NT LAN Manager (NTLM) (deprecated as of IBM Cognos BI version 10)
- SAP
- CA Siteminder
- IBM System Z Resource Control Facility (RACF)

As mentioned earlier, not all providers are available on all platforms supported by IBM Cognos BI.

**Custom Java Authentication Providers (CJAPs)**

In addition to the authentication providers supplied out of the box, IBM Cognos BI offers a Software Development Kit (SDK) for coding **custom authentication providers** in Java (CJAPs). A CJAP implements a well-defined Java interface and provides the functionality to read and search through security objects from the authentication source, authenticate based on various types of credentials, and possibly provide support for SSO. By using the SDK one can unlock the potential to attach to practically any type of authentication source and support SSO based on any given token.

There is also a special type of CJAP known as **Trusted Signon Provider** (TSP). A TSP is essentially a lightweight CJAP as it only implements one particular piece of functionality. A TSP acts as a proxy which sits in front of a full authentication provider and serves as a trusted (by IBM Cognos BI) party. It cannot exist stand-alone as it doesn't attach to any authentication source and doesn't appear as a Namespace in IBM Cognos Administration at runtime. It only
defines a Namespace object internally, based on configuration settings specified in IBM Cognos Configuration for the sake of interfacing. All a TSP does is consume a token from an HTTP request sent to IBM Cognos BI for authentication, applying whatever operations are necessary to deduce a user identity from that token and pass on the the original request to the secondary authentication provider after adding an additional token which is consumable by that secondary provider. It effectively translates a custom token into one which is supported by the configured secondary provider. The secondary provider will then handle that token as if it was passed to it from a trusted party which skips the initial verification step in that provider. Instead it will proceed directly to the validation of the passed user identity.

One example of a TSP is the Computer Associates (CA) SiteMinder authentication provider that IBM Cognos BI delivers out of the box. What the SiteMinder TSP does, is consume the proprietary encrypted SiteMinder cookie named SMSESSION, employs the SiteMinder API supplied by Computer Associates to decrypt the contents of it and then deduces the user identity from it. This user identity (which technically is a string) is put into the standard HTTP header REMOTE_USER before passing the request onto the secondary authentication provider. Of course, this secondary provider must support SSO based on the REMOTE_USER HTTP header. This is true for several of the authentication providers supplied out of the box with IBM Cognos BI. There are no users in the SiteMinder Namespace - those are only accessible through the secondary authentication provider configured for this authentication.

**Authentication conversations**

While it is straightforward to expect that a single request sent in for authentication contains all the required logon data (the Namespace to authenticate to along with valid credentials for that Namespace), this in fact is not true for all use cases. For an authentication provider, each request sent to it is treated independently but there are situations in which the authentication process is not a single step operation. Instead the authentication process could require multiple steps constituting a logical conversation similar to a relational database transaction. Consisting of several requests and responses exchanged between the client, the IBM Cognos entry point, the CAM_AsyncAAService and/or the handling authentication provider, the conversation as a whole either succeeds or fails.

This conversation concept allows for the return of qualified responses to the sender of the authentication request. The responses will either ask for additional information required to continue on with the authentication in a subsequent step or inform about errors that occurred preventing the continuation of the authentication process. The specific actions that the responses trigger with the client sending the requests will depend on the type of client and the type of response. These conversation concepts are described in the three scenarios mentioned in the Custom Authentication Provider Developer Guide, Chapter 2, Authentication Requests: flow scenarios.

If an authentication request doesn’t contain sufficient logon data to complete the authentication right away, the IBM Cognos BI authentication providers use a call-back mechanism that enables the authentication provider to request additional information from either the user or the system. The authentication provider returns what is called an exception and is similar to a Java exception in that the caller, in this case the sender of the authentication request, is responsible for handling
the exception. The caller is responsible for interpreting the exception and reacts to it by either sending another request containing additional information that will allow the authentication process to continue or by informing the user about possible errors.

There are three types of exceptions that an IBM Cognos BI authentication provider can return. These exceptions appear in trace level logs and form the basis for user facing dialogs rendered by the Presentation Service for HTML based clients. In the following descriptions of the exceptions, the actual labels and error codes that are found in the log files are provided in parenthesis.

- **UserRecoverable exception** (camAuthUserRecoverable, error -36)
  Signals the sender that the **end user/client** needs to provide data which should be attached to a new request. The exception returned to the sender contains specific information about the requested data. For HTML based clients, this implies that the sender will have to present a user interface to prompt the user for the missing information and re-send the request after adding the new information to it. The classic example is Presentation Service being employed to render a prompt page using an HTML form to gather the required data. Equally valid but less visible example is a web service client which will parse the response, render a custom authentication page and provide the entered credentials in it's response to the exception.

- **SystemRecoverable exceptions** (camAuthSystemRecoverable, error -37)
  Signals the IBM Cognos **entry point** that additional data is required which it must acquire **without** further interaction from the end user/client. These exceptions are used to implement SSO. All valid IBM Cognos entry point components handle these exceptions and read generic server variables and protected standard variables such as REMOTE_USER (Gateway and Dispatcher) and USER_PRINCIPAL (Dispatcher only) from the **local** (at the IBM Cognos entry point) environment. The additional data is attached to a copy of the original request (thus forming a new request) in an encoded and signed HTML form which is then automatically sent back to the authentication provider on the client's behalf - the actual client is not involved so the request won't appear in any client side HTTP tracing.

- **Unrecoverable exceptions** (camAuthUnRecoverable, error -38)
  Signals an error which cannot be remedied by any further action. This usually indicates some internal problem with the provider and renders authentication impossible - no further request must be sent for this authentication. The sender will inform the end user/client or log a failure as the authentication cannot proceed.

Overall this process implies an authentication conversation can have several round-trips of requests and responses between the authentication provider and the IBM Cognos entry point, client or end user in before the authentication finally succeeds or ultimately fails. This back and forth, which also involves CAM_AsyncAAService and CAM, is referred to as the **authentication dance**.

**Logon data**
The previous section introduced the concept of a conversation to gather sufficient logon data to process the authentication request. This section describes the actual types of logon data accepted by all the full authentication providers supplied with IBM Cognos BI.

There are four possible types of logon data in a request to the authentication provider
• Trusted Credentials (TCs)
  Credentials of this type have been created by an IBM Cognos BI authentication provider for a user during a previous authenticated session. In that session the user either explicitly asked to save his credentials for offline use or implicitly triggered the authentication provider to do so by saving a schedule. At this point the credentials provided to authenticate that previous session were encrypted and stored in the Content Store as part of the Trusted Credential object for that user. There is a single Trusted Credential object for each user and it can hold TCs from any of the defined Namespaces and associated authentication providers in an IBM Cognos BI system.
  For example, a schedule will store a reference known as a credential path to Trusted Credential objects. Technically those credentials can be a binary token or string(s) containing a username and password. Only the credential path is passed in requests - TCs never leave the Content Store.
  When receiving a set of TCs, an authentication provider will only implicitly verify their origin or consistency by calling a function which encapsulates the retrieval of the actual credential value form Content Store and decrypting it. To actually verify these credentials the provider simply runs through authentication against the authentication source.
  TCs may expire or become invalid. For example, if the password has been changed in the authentication source after they have been stored in Content Manager, the TCs must be renewed. Renewing TCs can be done explicitly in IBM Cognos Connection by the user who created them or, as of IBM Cognos BI version 10.1, they can be renewed automatically. The renewal process will update all the TCs against all of the Namespaces the user was logged into during the actual session.

• Credentials
  Credentials, also known as SDK Credentials, are sent by IBM Cognos SDK programs when authenticating to IBM Cognos BI. Technically they will be strings, likely a username and optional password. The credentials will be passed in the request in clear text unless the Secure Socket Layer (SSL) is used between the SDK program and an IBM Cognos BI Dispatcher. An authentication provider will verify the credentials against the authentication source by attempting authentication to it.

• HTML <form> fields - Username and Password
  If running Basic Authentication instead of SSO or credential based authentication, a user is prompted to enter their username and password. This implies Basic Authentication applies to interactive HTML based clients only. Once the user provides their credentials, the HTML pages generated by the Presentation Service will submit them in two predefined HTML <form> fields. The credentials will be visible (clear text) in the request sent to the IBM Cognos BI entry point. If the entry point is an IBM Cognos Gateway, the password parameter will be encrypted before the request is passed on to the CAM_AsyncAAAService. If the entry point is an IBM Cognos Dispatcher, the password encryption functionality is not provided. Using SSL encryption for the HTTP channel is recommended in sensitive environments to mitigate the risk of sniffing the passwords from the wire.

• SSO Tokens
  Authentication providers can also support SSO tokens. These tokens can be simple HTTP headers, CGI environment variables, protected server variables or cookies. Theoretically even
HTML `<form>` elements or any other element of a HTML request could be an SSO token. These will be covered later in the Single Sign-On to IBM Cognos BI section.

**Persisting authentication information in the sessions**

When serving a request, an authentication provider will use the provided logon data to authenticate to the authentication source attached to it. If authentication succeeds the session will be considered authenticated for the provided user identity to IBM Cognos BI and the provider will need to persist this state somehow. For this the authentication provider issues a visa for the Namespace the session was authenticated against.

A visa maintains the user's security context, that is the logon data (whatever credential or token was passed to IBM Cognos BI for authentication) and the user's identity (the groups and roles the user is a member of). The visa will, amongst other things, be used to derive credentials for other IBM Cognos BI services requesting authentication or to create Trusted Credentials that are stored with a schedule. The visa will be also added to an in-memory table managed by CAM which holds all active visas.

Since in a single session a user can authenticate to multiple Namespaces, there can be multiple visas for the session, one for each Namespace. All visas of a session get collected and managed in something called a Passport. Upon acquiring the first visa within a session, the Passport for this session is created and assigned a random generated identifier, the PassportID, which acts as the primary key for that Passport. The Passport is destroyed once the session logs off or expires.

The visas and the Passport are only kept in-memory, are managed by the CAM component and are never sent to any other service or component. Instead a reference to the PassportID, along with other non-sensitive session information, is signed and base64-encoded before being passed to the client in an HTTP session cookie named `cam_passport`.

Through the use of the `cam_passport` cookie, authentication is retained for all subsequent requests within the same client session. The cookie is sent to CAM_AsyncAAService for validation by the DispatcherService and if successful, the request will be processed. Any other outcome will trigger re-authentication.

By configuration, the cookie can be flagged secure so it will only be sent over SSL encrypted connections. In addition, the cookie domain and cookie path can be configured. By default, when no explicit domain or path is set, the cookie is sent back to the issuing server and path only. As of IBM Cognos BI version 10.1.1.0 (IBM Cognos 10 Refresh Pack 1), the `cam_passport` cookie supports the `HTTPOnly` flag which will prevent it from being read by scripts.

As mentioned earlier, sessions will timeout after a configurable amount of time if no validation of a passport has occurred to reset the idle timer. The default timeout value is 3600 seconds (60 minutes) and upon expiration, the Passport will be deleted and the reference stored in the `cam_passport` cookie is invalidated.
Authentication process

IBM Cognos BI clients such as browsers, SDK based applications such as BI Modeller and other products integrating with IBM Cognos BI security send requests calling for particular services. As described earlier in the Services, requests and sessions section, these requests will eventually hit an IBM Cognos Dispatcher Service. Authentication is usually implicitly triggered by the Dispatcher Service on any request. When it encounters an unauthenticated IBM Cognos BI session as indicated by the lack of the cam_passport cookie (the case for the initial request of a new session), the authentication process is triggered. When a session is already authenticated as indicated by the presence of the cam_passport cookie, the passport reference from the cookie is validated by calling the authentication process. If the session is found to be expired, the full authentication is triggered again. Only if the passport associated with a session is still valid will the Dispatcher Service continue processing the request and eventually assign it to an instance of the requested service.

However, there are some scenarios where authentication is invoked explicitly by one of the IBM Cognos services.

- Accessing a data source configured for authentication to an external Namespace (a namespace other than the Cognos Namespace) or configured to use a signon but where there is no signon available either due to authorization or lack of definition. In these cases authentication will be called explicitly by the Service executing the report though in most cases this will go unnoticed by the user unless it's required to prompt the user for credentials. This will happen whenever there is yet no visa for the external Namespace in the passport or if previous authentication to the external Namespace was by SSO which possibly did not require a credential suitable for database authentication but only some SSO token.
- When saving a schedule through Cognos Connection in a session that has been authenticated by SSO and thus some SSO token rather than a credential. It's common for SSO tokens to no contain passwords, depending on the authentication provider though this may invalidate the credential for storing it as a Trusted Credential. Authentication is invoked by Presentation Service to gather a credential (usually user name and password) sufficient for storing it as a Trusted Credential.
- User clicks the Log On link in IBM Cognos Connection to initiate authentication to yet another Namespace.
- The Logon/LogonAs methods get called by an IBM Cognos SDK client or another IBM Cognos BI service. For example, the scheduling functionality will have a service called the Monitor Service trigger the authentication to establish a session in order to execute something in background on behalf of a particular user.

Regardless of whether or not the authentication for a request has been invoked implicitly or explicitly, the Dispatcher Service instance will push a copy of the current request to a stack and forward the request to the CAM_AsyncAAService at which point the CAM component takes over. Once the authentication process is complete, the original request is retrieved form the stack and processed normally. When the CAM component receives a request for authentication, it first investigates the requested action as indicated by the <action> element in the <bus:CAM> section of the SOAP request. This action can be one of logon, logonAs or validate.
The validate action

The validate action is a lightweight operation compared to the other two actions. It is a quick call to validate an existing passport which is identified by a given passport reference. If the referred passport is still valid, CAM will issue a positive response to the caller which then proceeds processing. If the referenced passport is expired or non-existent, a negative result is returned which the sender of the request has to handle, usually by invoking the authentication process.

The logon action

This action calls for the overall logon process to start by investigating the general configuration of authentication. It is a generic actions where no prerequisites need to be met and no specific information is required to call for this action. This method is used to support anonymous authentication.

If the system is configured to allow anonymous access, then the sessions will become authenticated using a pre-defined anonymous user as there are no real anonymous sessions in IBM Cognos BI. An internal authentication provider is used to implement this functionality which is neither visible nor configurable. Some hard coded dummy user credentials are used, meaning no explicit credentials are required.

Anonymous authentication takes precedence, that is even if there is at least one active Namespace configured for named authentication, the anonymous user will get authenticated for logon actions. At the same time, if logon is called after the session has already been authenticated as anonymous, the authentication process will continue as it would for the logon action with anonymous access being disabled.

If anonymous access is disabled, the authentication proceeds to the logonAs action steps as if that action would have been called directly.

The logonAs action

A request sent to CAM which calls for the logonAs action skips the prelude offered by the logon action. This excluded anonymous authentication and specifically calls for authentication to an active Namespace.

The first step for CAM is to determine how many active authentication providers are available and thus how many Namespaces are available for authentication. When the CAM component started, it called each authentication provider to initialize and create an active instance. As of IBM Cognos 10, those instances are run in their own physical process called CAMLPSrv. For each instance of an authentication provider there will be one CAMLPSrv process. For example, in a system where three Namespaces have been configured but one fails to initialize, there will be two active authentication provider instances and two CAMLPSrv processes.

If there is more than one active Namespace available and none has been specified to be used for authentication yet, CAM starts an authentication conversation by throwing a UserRecoverable
exception which the client is requested to react upon. The expected response is to re-send the original request with an added indication of the Namespace to use for authentication. This is done by specifying the NamespaceID (from the Namespace configuration) of the Namespace to authenticate to, thus implicitly determining the authentication provider to use. The NamespaceID can be specified in an element of the `<bus:CAM>` structure in the SOAP request or as an HTML `<form>` parameter named **CAMNamespace**. For the re-sent request, the CAM action must be switched to logonAs because now there is an indication of which Namespace to use for authentication. For HTML clients such as browsers, this is done by script code embedded in the prompt pages generated by the Presentation Service. For any other type of client this has to be handled programmatically by setting the respective action in SOAP headers of the next request. This is done to ensure that the anonymous authentication process is excluded. For details about how this Namespace selection is handled for different clients, refer to the **Scenarios** section below.

If there is only one available Namespace or a Namespace has been selected explicitly by specifying its NamespaceID, the request is passed to the corresponding authentication provider. The provider will then process the authentication request according to its requirements.

The common authentication flow is described below. It applies to all authentication providers delivered out of the box with IBM Cognos BI (the **Cognos providers**) with the exception of the CA SiteMinder provider and the SharedSecret provider, both of which are TSPs. As mentioned earlier in the **Custom Java Authentication Providers (CJAPs)** section, TSPs don't implement authentication schemes, only proxy requests.

An authentication provider will first check whether sufficient logon data is available in the request to complete the authentication. Depending on the types of logon data supported by the provider, this will include Trusted Credentials used for scheduling or Credentials sent by an SDK client. Which type of logon data takes precedence depends on the provider but all Cognos providers check the logon data types in the following order:

- Trusted Credentials
- SDK Credentials
- SSO tokens
- HTML `<form>` variables

In a CJAP that implements a full authentication provider, some types of logon data might not be supported. One will need to check with the author of the CJAP as the SDK API does not enforce programmers to support all types of logon data.

In the case of Trusted or SDK Credentials being available in the request, the Cognos providers will not acknowledge SSO although it might be configured. Only if neither type of credentials are available will the Cognos providers check their configuration to see if SSO is enabled.

If SSO is enabled, the providers will return a SystemRecoverable exception which is used to start the authentication dance in order to obtain the trusted value of the SSO token. The number of round-trips (SystemRecoverable responses) required to provide sufficient information for the SSO
to be attempted depends entirely on the provider. For example, the LDAP provider will only use one SystemRecoverable exception to retrieve a server variable but the Microsoft Active Directory provider will go through several round-trips to exchange Kerberos tokens.

Once the provider has received sufficient data it will proceed with the SSO. The details of SSO will be covered later in this document in the **Single Sign-On to IBM Cognos BI** section. If the SSO succeeds, the authentication conversation will end with a response returning the cam_passport cookie and the success status to the client.

If there is no SSO configured, the Cognos providers will investigate the request for the presence of HTML `<form>` variables and if available, will attempt authentication based on those variables.

If the authentication based on the available logon data succeeds, the provider will issue the visa, pass the visa to CAM which will create or update the Passport for the session. In case of HTML clients where the Presentation Service relayed the request CAM will also issue or update the cam_passport cookie. For other clients the passport reference is returned as part of the SOAP response.

If there is no logon data available at all, a provider will respond with a UserRecoverable exception indicating the absence or invalidity of logon data. The sender of the request is responsible for handling this exception and the expectation is that the request will be re-sent after attaching one of the supported types of logon data.

Other errors such as connection issues to the authentication source and errors returned by the authentication source, will be reported back in an Unrecoverable exception to the sender of the request. The sender of the request is responsible for handling this exception.

**Scenarios**

**Scenario 1** - A browser client accesses the IBM Cognos Connection default page in a new browser session by reaching out to an IBM Cognos CGI gateway deployed to some web server. There are multiple Namespaces configured, anonymous is disallowed and no SSO has been set up.

- Accessing IBM Cognos Connection implies calling Presentation Service which is responsible for rendering the HTML based GUI. The Dispatcher Service will notice that the session is not yet authenticated and will push a copy of the request to a stack.
- Next the Dispatcher Service attaches a CAM action of **logon** to the request and sends the request off to CAM_AsyncAAService. CAM finds out that there’s no anonymous access allowed and that no Namespace is being selected for authentication. Consequently, it will throw a UserRecoverable exception. Part of the exception response is a `<promptInfo>` element which contains a caption like “Please select a Namespace for authentication” and all the possible values to select from in this case, all the Namespace names and NamespaceIDs of all initialized Namespaces.
- The Dispatcher Service is the sender of the request so it has to handle the UserRecoverable exception. It knows about the client being a browser (based on the fact that Presentation
Service was called originally) and will employ a local instance of Presentation Service to render an HTML prompt page based on the data received in the `<promptInfo>` element. If there is no local instance of Presentation Service available, the exception cannot be handled and is instead passed through to the client in raw XML. Browser clients will typically display the XML which at this point indicates a configuration error, since Dispatchers serving as entry points must run an instance of Presentation Service.

- Once the user selects a Namespace by name on the generated prompt page, a hidden HTML form submits (using the HTTP POST method) another request to which two parameters have been added. The first parameter, **CAMNamespace**, contains the NamespaceID of the selected Namespace and the second parameter, **h_CAM_action**, is set to **logonAs**.
- This time the request will be assigned to the actual authentication provider behind the selected Namespace by CAM. The provider will investigate the request and fail to find any supported logon data. Since SSO has not been set up, the only choice therefore is returning another UserRecoverable exception asking for any logon data. Because there were no Trusted or SDK Credentials in the request and SSO is disabled, the Cognos providers will ask for `<form>` variables containing a username and password to be provided on the next attempt. Therefore, as part of the exception response, a `<promptInfo>` element which contains the suggested HTML forms and labels is returned.
- Again Dispatcher Service will handle this exception and, because it knows the request came in through Presentation Service, will employ the local instance of Presentation Service to render yet another HTML prompt page using the information from the exception's `<promptInfo>` element to create an HTML form. On the prompt page, a user can type in their username and password. When the user submits the form, the username and password values are mapped to the **CAMUserName** and **CAMPassword** parameters respectively and the updated request is sent by HTTP POST to CAM_AsyncAAService through Dispatcher Service which passes the request to the previously selected authentication provider.
- This time the provider finds FORM variables containing credentials and thus has sufficient logon data to attempt actual authentication to the external authentication source.
- If the authentication succeeds, the Dispatcher Service retrieves the original request from the stack and adds the updated session information, which now contains the new visa issued by the provider, and continues processing. This scenario implies forwarding the request to an instance of the Presentation Service (which renders the HTML responses for the client) based on Dispatcher routing concepts. If the authentication failed due to invalid credentials, the provider will have returned another UserRecoverable exception asking for valid credentials and the steps mentioned above repeat from the point where the user is prompted for username and password.
- If an unexpected event such as a failed connection to the provider happens, an Unrecoverable exception will be returned which will lead to the Dispatcher Service asking the Presentation Service to render an error page thus ending the authentication process.

This is one of the most typical scenarios. It demonstrates the conversation concept and that a single authentication can contain one to many exceptions, which although carrying error codes, are actually part of a successful authentication. This is important when troubleshooting authentication.
Scenario 2 - A browser client accesses IBM Cognos Connection by reaching out to an IBM Cognos CGI Gateway deployed to an IBM Web Server (IHS) in a new browser session. A single Namespace has been configured for SSO. Anonymous access is disabled.

- Accessing IBM Cognos Connection implies calling Presentation Service which is responsible for rendering the HTML based GUI. The Dispatcher Service will notice that the session is not yet authenticated and will push a copy of the request to a stack.
- Next the Dispatcher Service attaches a CAM action of logon and sends the request off to CAM_AsyncAAService. CAM finds out that there's no anonymous access allowed and that no Namespace is being selected for authentication. However, since there is only one active Namespace, the request is passed to the only initialized provider.
- The provider is configured for SSO, therefore it will identify the required SSO token. Depending on the type of token (covered later in the Single Sign-On to IBM Cognos BI section), the provider would continue to the next step straight away or, as we will assume for this example, return a SystemRecoverable exception asking that a token value be provided by the IBM Cognos entry point. The reason for returning the exception is that even if the SSO token is available as part of the request, the provider will not take the token “as is” for security reasons. To mitigate spoofing, the provider will only accept SSO tokens to be sent by trusted parties such as other Cognos system components.
- The Dispatcher Service notices the exception of type SystemRecoverable. Based on the information so far, the Dispatcher knows it is not the entry point - the request was forwarded to it from a Gateway. Therefore it is not supposed to handle the SystemRecoverable exception, only the entry point is. If however the Dispatcher would have been accessed by the browser client directly, it would have been the entry point and thus would have handled the exception. However, in this scenario, it assigns a HTTP response code of 599 to the response and passes it back to the entry point which in this example is the IBM Cognos CGI Gateway deployed to IHS. The Gateway catches the exception, deduces the requested token from it's local environment and sends another request. This happens automatically without client interaction. Therefore this second request will not be visible to the client and it won't appear in HTTP level traces of the communication between client and the web server. The request being sent is a copy of the original request with the retrieved token value attached in a special encoded and digitally signed HTML form parameter named CAM_SecurityBlob.
- The second request via Dispatcher Service hits CAM_AsyncAAService and is passed to the authentication provider again. The provider unwraps the CAM_SecurityBlob FORM variable and attempts the SSO based on the value retrieved.
- If the authentication succeeds, the Dispatcher Service retrieves the original request from the stack, adds the updated session information, which now contains the new visa issued by the provider, and continues processing. In this scenario, this implies forwarding the request to an instance of the Presentation Service which renders the HTML responses for the client based on Dispatcher routing concepts.
- If the authentication failed due to invalid credentials, the provider will have returned a UserRecoverable exception asking for valid credentials. At this point, since SSO failed, falling back to get the logon data from FORM variables is assumed and the exception will ask for a username and password only. Since the Dispatcher Service knows the client to be a browser because Presentation Service was called originally, Dispatcher Service will employ the local
instance of Presentation Service to render a login page and the authentication switches over to an interactive authentication similar to that described in Scenario 1.

- If an unexpected event such as a failed connection to the provider happens, an Unrecoverable exception will be returned which will lead to the Dispatcher Service to asking the Presentation Service to render an error page thus ending the authentication process.

For this second scenario the main takeaway is that SystemRecoverable exceptions are handled at the entry point only and won't make it back to the client. Only by tracing authentication in IBM Cognos BI or capturing HTTP traffic between the web server and the IBM Cognos BI application tier will those requests will become visible.

**Non-Browser/SDK based clients**

Both of the scenarios just presented cover browser clients only because they offer the possibility of user interaction. Thick client applications such as IBM Cognos BI Modeler employ an internal version of IBM Cognos BI SDK which requires interactive, user-driven authentication to be coded explicitly. Those SDK based clients therefore will often use a two-step approach to drive the authentication with the IBM Cognos BI back end. First, they employ an embedded Internet Explorer browser control to run the authentication. This provides functionality available to browser clients such as customized login pages and third party authentication proxy support but also allows the thick client to retrieve the value of the cam_passport cookie once the session has been authenticated. Then, subsequent communication with IBM Cognos BI between the client and the IBM Cognos BI Dispatcher URI is done leveraging SDK functionality which technically reflects in sending SOAP messages back and forth with the cam_passport cookie value embedded in the SOAP requests and serving as the means for authentication.

Other IBM Cognos products such as IBM Cognos TM1 and IBM Cognos Planning allow for security integration to IBM Cognos BI as well. They therefore act as clients. Technically these products employ some software components which again simulate or behave like a browser client to establish an authenticated session first. For this they call special Presentation Service templates such as *bridge.xts* which allows for passing the cam_passport cookie to external applications (refer to the *IBM Cognos Software Development Kit Developer Guide* for details). Because they call the Presentation Service, most, if not all, functionality of the standard IBM Cognos BI authentication (customized login pages, SSO and support for reverse (authentication) proxies) is available to them as well. Once the cam_passport cookie is retrieved from the established session, subsequent access to IBM Cognos BI again is based on SDK which means SOAP requests sent to the IBM Cognos BI Dispatcher URI directly.

Pure SDK clients which don't leverage the approach of simulating HTML clients have to rely on processing SOAP messages directly. These clients will call functions exposed by the API to run authentication and passports will travel the network embedded in SOAP requests and responses. Due to the increased security concern this poses, it is advised that communication be secured, at a minimum, by SSL.
Single Sign-On to IBM Cognos BI

Concepts

Gaining access to IBM Cognos BI without logging in explicitly is commonly referred to as **single sign-on (SSO) to IBM Cognos BI**. A more general definition of SSO describes it as *the process of gaining access to multiple independent, but related, systems based on a single login*. For IBM Cognos BI this means that a user must have been authenticated to an identity assigned to the HTTP session prior to accessing IBM Cognos BI within that very same session. That authentication must have been completed by presenting credentials to an external-to-Cognos security system. The security system might provide the identity and/or some sort of credential information suitable for achieving SSO to other systems, usually in the form of what commonly is referred to as an **SSO token**. Typical candidates for such security systems are authentication proxies such as IBM Tivoli WebSEAL, Oracle Oblix, Computer Associates SiteMinder or any other software or hardware solution which can authenticate a HTTP session and persist that authentication in a token.

Based on this, the main concept for SSO support in IBM Cognos BI is that **IBM Cognos BI consumes SSO tokens from a single request sent to it**. How the token got into the request is not important and the IBM Cognos BI product is agnostic of it. What happened before the request hit an IBM Cognos BI entry point, how many previous hops of a multi-system SSO have taken place and how many security systems are involved is irrelevant. Only when the request hits an IBM Cognos BI entry point does the processing of SSO tokens start. This defines a distinct border between any third-party security system and IBM Cognos BI and this border is of particular importance when involving multiple software vendors for assistance.

This design, which only considers a single request, allows for third-party products that employ HTTP 30x return codes or scripts to redirect the client to a different URL, run through some login logic and return to the resource URL requested originally. To IBM Cognos BI this kind of prelude conversation must be (and usually is) transparent as requests are processed by IBM Cognos BI independently. In fact many **seamless authentication** scenarios described as **SSO to Cognos** actually consist of multiple hops, each involving some form of authentication or trusted sign-on.

A typical scenario is a user logging in to their machine where they get authenticated by the operating system (OS) security. Next, the user opens a browser and accesses a protected resource on a web server. Access to that resource is governed by web server security. Ideally there is some SSO from the operating system to the web server security. If not, the user would get prompted to authenticate to the web server’s security layer. If this protected resource happens to be an IBM Cognos BI Gateway, another SSO hop would be from the web server security to IBM Cognos BI. In this scenario, there are two hops for overall SSO – from the OS to web server and then from the web server to IBM Cognos BI. This assumes the SSO to IBM Cognos BI is based on some information provided by the web server security but it could be based instead on the operating system’s security if a token was available.

In order to successfully achieve SSO for IBM Cognos BI, all that matters is the SSO token and whatever information is available from that SSO token (in particular the credential and identity) to be consumable by any IBM Cognos BI authentication provider. As mentioned before, IBM
Cognos BI supports two types of authentication providers - the full providers which attach to some authentication source and authenticate sessions and trusted security providers (TSPs) which serve as a proxy but cannot authenticate sessions themselves. From this it can be seen that for SSO, it's the full providers' support for specific authentication schemes and token types which determine the available configuration options.

The out-of-the-box full providers for IBM Cognos BI support only two authentication schemes for SSO.

- **Authentication based on credentials** - The provider receives credentials in the form of some SSO token which it presents to its configured back end system unchanged. This is known as **pass-through**. In fact, it has to be assumed that the provider's back end authentication source either is the same security system which issued the SSO token or at least one which can consume the SSO token in use. Examples are Kerberos tokens for the Active Directory provider, the SAP cookie for the SAP provider and the CA SiteMinder cookie for the SiteMinder provider.

- **Trusted sign-on based on identity information (identity mapping)** – In this context, trusted sign-on means that the provider will authenticate the session based solely on a provided identity, no explicit credentials are required. The authentication provider will trust the identity if it was provided and assured by a trusted source. The only trusted sources for the Cognos providers are the IBM Cognos BI Gateway and Dispatcher entry points. For trusted sign-on, no re-authentication with any security system takes place, rather a simple validation of the identity takes place in the form of a look-up in a configured back end authentication source. If the look-up is successful, that is considered sufficient to complete the authentication. This scheme is less secure as it bears the risk of forging identity information or unintentionally granting access because the same identity exists in two systems. For example, a user "Bob" is authenticated by Windows and that identity is also provided to an IBM Cognos BI LDAP provider attaching to Tivoli LDAP. If the identity "Bob" exists in the Tivoli LDAP, IBM Cognos BI authenticates it. Whether or not the identities refer to the same actual user is not guaranteed at this point because the two security systems in this example (Windows and Tivoli LDAP) aren't necessarily related. Even though the assignment of identity cannot be guaranteed by identity mapping, on the other hand it allows for greater flexibility in achieving integration of technically disjointed systems (in this example a stand-alone LDAP and a Windows domain) when it's clear that the identities are assigned to the very same user. Identity mapping is supported by the LDAP provider through its external identity mapping feature and the Active Directory provider when configured for identity mapping SSO mode.

Information used in either SSO authentication scheme is passed in SSO tokens. Technically there are three types of tokens to be used for the two schemes.

- **Cookies** – Cookies are strings which get sent along with HTTP requests in the standard HTTP header named "Cookies:". The predominant use for cookies is to carry credentials, although theoretically they can hold identities as well.

- **HTTP headers** - Any arbitrary HTTP header is a clear text string which is sent along HTTP requests. The HTTP protocol defines standard headers but clients or servers can also add their own self-defined headers.
• **Server variables** - A server variable is a string variable which only exists in the local environment of the web server or Java application server and is not transmitted as part of an HTTP request. A special subset of server variables that applies to web servers only is the set of variables which gets provided to CGI web server extensions. This set differs by web server but its variables are commonly referred to as **CGI environment variables**. Access to those variables is only possible through code executed in the context of the actual web server. This is how CGIs or other server extension modules operate. That code can then submit the retrieved values on its own using proprietary techniques. Server variables are usually used to provide identity.

From the above one could imply that, depending on the tokens and authentication schemes supported by the Cognos providers, SSO support would be limited to a small set of options. While it is correct that the Cognos providers only support a defined set of tokens and schemes, the IBM Cognos BI SDK provides an Application Programming Interface (API) that allows one the ability to create a custom authentication provider (CJAP) to supplement the existing providers and extend the support to basically any type of SSO token.

Creating a full provider CJAP is an endeavor one would take on only when support for a specific authentication source is required. The task is complex and requires a considerable amount of knowledge across several subjects (how the back end authentication source works, the HTTP protocol and advanced Java programming to name a few), effort and skill. Yet for SSO, which is the focus here, usually one is looking to simply close the gap between a token provided by some security system and an authentication source already supported by an IBM Cognos BI authentication provider. This can be achieved by creating a trusted sign-on provider (TSP), which is a much less complex task than creating a full provider and can usually be done in a fraction of the time it would take to create a full provider CJAP.

Although only full IBM Cognos BI authentication providers can authenticate a session for IBM Cognos BI, a TSP can consume all three of the SSO token types. A TSP by design acts as a proxy - it receives and processes the request and passes it on to a full authentication provider. The processing can include consuming any of the token types introduced above and can also add tokens of any type. The API provided as part of the IBM Cognos BI SDK for authentication providers defines the required methods. For example, a TSP can consume one token and produce another different one, regardless of type and authentication scheme required. It can even go so far as to prompt the user for additional information to implement, for example, a 2-factor authentication, where a user would need to provide something they **know** (such as a password or pin) plus something they **own** (such as a keycard). This is why TSPs are very powerful and constitute the ideal means to enable SSO.

As shown, IBM Cognos BI can support almost any SSO integration although some may require custom programming depending on the token being used.

It is important to note that there is currently no support for passing **authorization** information during SSO. Authorization is completely proprietary in IBM Cognos BI and cannot be integrated with other solutions without further use of the IBM Cognos BI SDK to programmatically manage authentication. Theoretically one could include the required SDK calls to manage authorization
in the processing of authentication, but this can only be done in a CJAP since the Cognos providers don't allow for extension. A CJAP requires custom coding that is normally specific to an installation and is not supported by IBM Cognos BI. Intermingling authentication and authorization isn't considered a best practice either. To handle requirements such as automatically granting authorization to a user after they have been authenticated should be handled in an **Authentication Listener**. The Authentication Listener interface allows registering custom functions which get triggered based on events such as successful authentication or session expiry. Refer to the *IBM Cognos Custom Authentication Provider Developer Guide* for details on Authentication Listeners.

**Single Sign-Off** is supported as of IBM Cognos BI version 10.1 through a setting which allows for re-directing the client to a specified URL upon him explicitly logging off. The redirection does not apply to session expiry though. For details regarding configuration, refer to the section titled *Customizing the IBM Cognos Connection Login Page* in the *IBM Cognos Administration and Security Guide*.

For more detailed descriptions of the Cognos providers that delivered with the product and their configuration, refer to the section titled *Configuring IBM Cognos Components to Use an Authentication Provider* in the *IBM Cognos Installation and Configuration Guide*.

**Technical implementation**

A full IBM Cognos BI authentication provider will typically involve the following high level steps for SSO,

- **consume** - Implicitly or explicitly verify the token content. Tokens get verified implicitly by decoding/decrypting them either by code within the provider itself or by employing third-party APIs. If the token decrypts without error the authentication process will commence. Explicit verification can happen by parsing the contents of the token or verifying signatures. Either way, some credentials will be available from the token after this step.
- **validate** - Either present the credentials to the configured authentication source for validation and thus obtain an identity or validate the deduced identity from the token with the configured authentication source.
- **authenticate** - Authenticate the session with IBM Cognos BI if the previous step was successful.

For TSPs the high-level steps are only slightly different,

- **consume** - Implicitly or explicitly verify the token content as described above for a full authentication provider.
- **transform** - Deduce whatever credential/identity information is required from the token and possibly apply transformations to it. By design, a TSP passes authentication on to a secondary full authentication provider. Therefore the TSP may have to construct a new token which is of a type that is directly supported by the secondary full provider.
- **pass-on** – Populate the token to pass on and hand over the authentication process to the secondary full authentication provider.
The full authentication providers delivered with IBM Cognos BI for SAP, IBM Cognos Series 7, LDAP, Microsoft Active Directory, Microsoft NTLM and RACF follow the described rundown and will be discussed in detail later in this document. The CA SiteMinder provider, which is a TSP, will also be discussed in detail.

Both provider types share the step in which a token is consumed. Earlier we introduced the three token types - Cookies, HTTP headers and Server variables. While the support for particular token types differs by provider and the specific steps to consume the supported token types vary, one common and important aspect of all SSO tokens is their trustworthiness. The goal of SSO is to take on existing information and trust it for a subsequent authentication therefore it is essential to have an understanding of the trustworthiness of a particular token type. If a token type is easily forged or altered, the security level of the SSO solution is lowered.

IBM Cognos employs some unique techniques to ensure trustworthiness for tokens passed to it. Before exploring these techniques further, we first need to cover some of the technical details about the supported token types.

**SSO token types in depth**

**Cookies**

Cookies are the most convenient tokens as their handling is based on standardized methods and practices. Like all content in an HTTP request, they are transmitted in clear text by design of the HTTP protocol but there are many security features included in the standard for handling cookies. Items such as configurable expiry, controls to limit the sending of cookie to servers at the client side or the enforcement of SSL encrypted HTTP channels are just some examples of the security features. These standards are adopted by almost all HTTP client and server products to protect cookies against tampering, injection, highjacking and forging. Since cookies are transmitted via HTTP and can also be stored to the local file system by browser clients, cookies are usually encrypted and encoded to further tighten security on them. In state of the art encryption schemes, a feature known as signing is used to ensure the integrity of transmitted contents which allows the recipient the ability to determine if the contents of the cookie was altered on its way from the server. All of this makes cookies the most trustworthy token type to use for SSO and they are supported for that purpose by many commercial solutions.

An IBM Cognos BI authentication provider reads cookies directly from the request and in case of a TSP, can even add cookies to the request. Several Cognos providers support cookies and leverage third-party APIs to decrypt their contents and deduce the required information from them.

**HTTP Headers**

An HTTP header is a string which is included in the header section of a HTTP request. There are several pre-defined headers in the HTTP standard and the sender of a request can define and add it's own arbitrary headers. HTTP headers can be added to a HTTP request by either the client or the server. Typically a client will send a set of headers to the server to control aspects of request processing or to supply information about the expected syntax, format or encoding of the server's
response. The HTTP headers sent by a client to the server are referred to as request headers while the HTTP headers sent by the server to the client are referred to as response headers.

HTTP headers can carry either identity information or credentials. They are accessible by servers and clients as well as parties such as routers, proxies and firewalls that handle the HTTP request as part of the transmission. Therefore HTTP headers are considered the least trustworthy token type because it's technically very easy to forge, inject or tamper with them. There is no means to distinguish between a forged value and an original value. Headers are typically not signed or encrypted although technically this would be possible. Since they are relatively easy to tamper with, HTTP headers are typically a last resort to achieve SSO, typically by identity mapping.

Server Variables

Server variables, sometimes referred to as CGI environment variables, are variables which only exist on a HTTP server. While this usually implies web servers, this is not entirely true. Java application servers contain a HTTP server component because the SOAP protocol commonly used in Java Platform Enterprise Edition (J2EE) uses HTTP as the transport protocol.

Every web server or Java application server supports a different set of server variables which can be categorized into three main classes.

- Simple variables which get set by the server only. An example of this type of server variable is AUTH_TYPE which indicates which authentication method is supported by the server.
- Protected standard variables which get populated by the server as a result of successful authentication to it only. The standard variables are REMOTE_USER and USER_PRINCIPAL.
- So-called “meta-variables” which get populated based on request headers. A client can send HTTP request headers along with a request to the server. These request headers will be reflected as special server variables (the meta-variables) in the context of the server automatically. This is done by converting each request header to upper case, replacing all occurrences of “-” (dash) by “_” (underscore) and prefixing the header name with the string “HTTP_”. For example, if the client sends the HTTP header “My-User” along with the request to the server, the server will automatically create the a meta-variable “HTTP_MY_USER”. This meta-variable will only exist on the server and as one would expect, its value will be the request header value of “My-User”. Effectively this means that a request header can be obtained by server code in two ways, either as a raw HTTP header fetched from from the request directly or as a meta-variable from the server variables.

Depending on the server, some or all of these variables contribute to an environment which is provided to programs and executables run in the server's context. Web servers typically provide variables from all the above categories to its extensions while Java application servers only know protected standard variables and meta-variables. There are no simple variables for Java application servers due to architectural differences when compared to web servers.

As mentioned before, server variables are accessible only by code executed in the context of the server. They don't get sent along in HTTP requests automatically. Rather, code which has access to the server variables must implement its own means of passing the retrieved information to other
software components, local or remote. We'll touch on this again shortly but first let's understand what environment is provided to code deployed to web servers.

Web servers usually provide two interfaces for programming executables which can be deployed to the server to extend the server's functionality. The executables are known as server extensions. The first interface is the standard known as the **Common Gateway Interface (CGI)** and executables that implement this interface are known as **CGIs**. CGIs can be implemented in many different programming languages and are supported by the almost every web server. The CGI standard is defined by several RFCs which make them versatile but not necessarily efficient. The second interface that most web servers offer is specific to the implementation of the web server. Microsoft's Internet Information Server (IIS) provides the **Internet Server Application Programming Interface (ISAPI)** and the Apache web server supports an interface known as a **MOD**. These interfaces overcome the limitations of the CGI interface with regards to performance and integration with the actual web server design.

Executables implementing the Common Gateway Interface implicitly define the set of variables available to CGIs based on the CGI standard. This set is sometimes referred to as the **CGI Environment**. There is a defined API call in the CGI standard to read variables from the CGI environment as well, the call named **GetEnv()**. It provides the value of the requested server variable to the caller. Using this standardized approach CGI programs can read every simple server variable, the protected server variables and even the meta-variables which are in fact the representation of the HTTP headers of the received request.

In contrast to the CGI environment and the access provided to it for CGI programs, the environment provided by web server specific interfaces depends solely on the vendor of the web server. Usually the standard CGI environment variables are supplemented by a set of proprietary variables which only apply to a specific server. Access to the server variables is provided by a function call in the proprietary interface - usually named **GetEnv()** (or something very similar) to imply the same functionality as the respective function call in standard CGI programming. However, unlike the standard function for CGI programs, these proprietary implementations usually only mimic the CGI standard implementation. The implementations may provide a subset of the server variables only or extend or limit support of particular server variable categories. As an example consider the Microsoft IIS implementation for the ISAPI interface. Its **GetEnv()** call does treat meta-variables different, using different substitution rules to overcome alleged CGI standard definition gaps. In practical applications this leads to inconsistent results when considering different web servers.

Another common difference is that in proprietary interfaces, protected server variables are not accessible by **GetEnv()** calls but particular function calls only. For example, there is a specific call to retrieve the value of the REMOTE_USER variable in the ISAPI interface and in the interface for Apache MODs. This is to increase security and correctly guard access to this important category of variables.

In summary, web server extensions are provided with simple, protected and meta-variables. Depending on the interface though (CGI vs. proprietary) access to them requires different code.
While CGI programs are portable and will work with any web server which supports the CGI standard, this does not apply to server extensions using proprietary interfaces.

For more information about server variables for the Apache and Microsoft IIS web servers, consult the Resources section at the bottom of this document.

It was mentioned earlier that Java application servers also implement HTTP servers but may have limited features. This triggers the question on how server extensions and their environment are implemented for Java application servers.

Without going into detail on how an application is deployed into a Java application server, it is sufficient to state that executable code can be deployed to a Java application server in two forms - a Servlet or a Java Bean. Java application servers run executables in something called a container and there are specific containers for Servlets and Java Beans. These containers provide the executable with an environment that has been defined in the J2EE standard. For now, this document will just focus on Servlets.

The concept of an environment is different in Java application servers due to architectural differences with web servers. The Servlet container environment is more comparable to the environment provided by the web server specific interfaces such as ISAPI and MOD than it is to the CGI environment. It does not provide any type of server variables. Instead Servlet containers provide access to the information contained in most simple server variables using calls defined in the Servlet specification. In addition, they provide access to the actual request received from the client which enables Servlets to access request headers directly instead of reading meta-variables. Last, due to the lack of server variables, a mechanism for providing the two protected server variables that are used in SSO (REMOTE_USER and USER_PRINCIPAL) is required. This mechanism is provided through dedicated standard function calls.

As an example, consider a Servlet which would like to obtain information about what scheme of authorization data was sent along with a request to the server. This information on web servers is available in the simple server variable AUTH_TYPE. Since Servlet containers do not offer server variables, the information has to be obtained through a call defined in the Servlet specification. If however the Servlet would have looked for REFERER information, which is an optional HTTP header, it could have read the information from the request directly.

To evaluate the level of trustworthiness associated with server variables one must be aware of how external applications that access the server variables handle them. As stated before, server variables do not leave the server. They don't get attached to HTTP requests, they can only be accessed locally on the server. That implies that if some software wants to use information contained in server variables for implementing SSO, this information must either be processed on the server itself by deploying a piece of software to it or the information must be passed outside of the server. Since there are no defined standards on how to pass server variables outside of the server, every software solution implements its own approach to this. The trustworthiness of a server variable is fixed but if an external application is forced to pass the information retrieved from a server variable outside of the server, then the trustworthiness of the solution is dependent on the trustworthiness of the application.
To summarize server variables, one can state:

- Server variables of type meta-variable are insecure and should not be used to implement SSO. This is because meta-variables are representations of request headers, so anyone can send in a value for the respective meta-variable with a forged HTTP header attached to an otherwise valid request.
- Protected server variables by design are very trustworthy and are ideal candidates for being used in SSO solutions. Only when a user has been authenticated by the server’s security layer, these variables will be populated. Depending on whether or not the information read from them is passed to remote software applications, protected server variables should be used as a best practice.
- Simple server variables are trustworthy as well, however their level of security is weaker than that of protected server variables as have the potential to be forged by a malicious user who has local access to the web server itself. In addition, the possible transport of information retrieved from them to other software components influences the overall trustworthiness of the solution.

**The Cognos handling of SSO tokens**

The previous section delivered a rather technical view on SSO tokens. It was mentioned that IBM Cognos BI uses a unique approach to handle the consumption of SSO tokens, an approach that is based on the Cognos architecture. This section will describe that approach.

The IBM Cognos BI product documentation and the *Custom Java Authentication Provider Developer Guide* describe leveraging external tokens for SSO. They touch on cookies very briefly and then refer to what are known as Environment Variables. Further to this, there is a distinction between simple environment variables and trusted environment variables.

Simple environment variables are described as variables that store information “from the entry point”. No further details are provided regarding the origin of the variables - if they are originally provided by server hosting the entry point or are simply relayed by it to the authentication provider. In contrast, the trusted environment variables are described as variables that are “signed by the Cognos entry point Gateway”, implying that the information provided by trusted environment variables is trustworthy because the IBM Cognos BI Gateway signed the variable to protect it from being tampered on its way from the Gateway to the IBM Cognos BI authentication provider. With simple environment variables not being signed, it can be stated that simple environment variables are not necessarily trustworthy. While not explicitly mentioned in the documentation, trusted environment variables also contain information “from the entry point” and more importantly, they are provided by every supported entry point.

IBM Cognos BI authentication providers make use of an API which defines calls by which an authentication provider can either read a simple or a trusted environment variable or, if required, cookies. The *IBM Cognos Custom Authentication Provider Developer Guide* describes the Java version of this API for those interested in the details. What requires further explanation is the handling of the environment variables and the role of the IBM Cognos entry point for retrieving them.
Earlier in this document, we saw that for IBM Cognos BI there are two valid entry points. The first, and preferred, entry point is an IBM Cognos Gateway deployed to a web server. Alternatively, a Servlet Gateway or a Dispatcher, both deployed to Java application servers, can also serve as an entry point.

The IBM Cognos Gateways that are deployed to web servers exist in three different implementations: CGI, ISAPI and MOD. These implementations leverage the web server extension interfaces introduced earlier in the section **SSO token types in depth** and they represent the IBM Cognos BI architecture in the web tier of a classical three-tier architecture. IBM Cognos BI interfacing the web tier is important because handling authentication for a web application is usually done in the web tier and as such, the web server plays an important role. Many commercial products for implementing web application security such as IBM Tivoli Access Manager through its WebSEAL component or products such as CA SiteMinder or Central Authentication Service (CAS) are deployed to the web tier and sometimes even interface with the web servers directly. In addition, web servers deliver an authentication layer as a built-in feature. The predominant use case for implementing SSO to IBM Cognos BI includes some authentication system running at the web tier, usually providing some SSO token to the web server using a cookie or a server variable such as REMOTE_USER. This is why having an IBM Cognos Gateway deployed to the web server is actually considered a best practice for SSO configurations (although strictly speaking the IBM Cognos Gateway is not a must-have component in the architecture of an IBM Cognos BI system) as the IBM Cognos Gateway allows access to server variables on the web server which is often required for SSO integration.

However, IBM Cognos BI also supports entry points deployed to the application tier - the Servlet Gateway and the Dispatchers. They provide similar functionality regarding the Cognos authentication process by allowing access to server variables on Java application servers. Application tier entry points are used for SSO scenarios where the authentication happens at the application tier only - for example when requests get forwarded by a proxy in the web tier or when the SSO token is only available at the Java application server.

Common to either type of entry point is the way they handle SSO tokens. At this point, based on the backgrounds described earlier in the **Technical implementation** section and the concepts introduced so far in this section, it's time to match the technical view with the Cognos view on SSO tokens.

An IBM Cognos entry point will receive an HTTP request which is subsequently passed to the authentication provider. Part of that request are request headers and cookies. While cookies provide a trustworthy means for SSO tokens, as explained earlier HTTP headers, by default, are not actually trustworthy. Now recall that Cognos explicitly distinguishes between simple or trusted environment variables. From this, one can conclude that when a Cognos provider asks to retrieve a simple environment variable it means reading a HTTP header from the request “as is”. The information obtained from the header is not trustworthy because the request headers could have been forged. This may make the wording in the documentation “from the entry point” seem ambiguous because the request header originates from a client. What's meant is the value which is received by the entry point.
We will now turn our focus on trusted environment variables. The implication of “trusted” in this Cognos context is that if an authentication provider, which can be physically separated from the entry point, accesses a trusted environment variable, the provided values are exactly the same as they are locally on the entry point. If remote software components are involved, the value of a variable which exists on the server hosting the entry point must be transmitted securely between remote components. The need to transmit securely combined with the technical details about server variables, the authentication conversation concept covered earlier in the Authentication conversations section and the entry point considerations identifies the two main concepts of the IBM Cognos BI approach for implementing this trustworthiness.

1. The Cognos entry point is deployed to the web server or application server. This allows for access to server variables on that server by Cognos code. The Cognos Gateways therefore can provide server variables to other IBM Cognos BI components.

2. The challenge of securely transporting retrieved server variables to a possibly remote instance of CAM is solved by combining two techniques. The entry point code digitally signs the server variable values retrieved from the server environment so that a Cognos authentication provider can verify that the information it's receiving was not altered or damaged and in fact originates from a valid Cognos entry point. For transporting the server variable values from the entry point to CAM and onto an authentication provider, the authentication conversation concept is employed.

Let's look at this in detail. A Cognos authentication provider requests a trusted environment variable through some API call. This call triggers CAM to investigate the received authentication request for a HTML `<form>` parameter called CAM_SecurityBlob. In this parameter a Cognos entry point would provide a response to a SystemRecoverable exception. If this parameter is not found, the call to retrieve the trusted environment variable returns an indication that a SystemRecoverable exception is required to “prompt the system for information”. The authentication provider code reacts to this by throwing a SystemRecoverable exception, attaching the name of the requested server variable in the exception's prompt info. By employing a proprietary HTTP response code of 599, the Cognos entry point which originally relayed the request to CAM through the authentication conversation is informed that it has to handle that exception. Handling means the entry point will investigate the response for a specific SOAP element named SecurityBlob. This element contains a base64 encoded binary representation of the name of the server variable the authentication provider is requesting and was previously specified in the SystemRecoverable exception's prompt info. The entry point code now employs the web server's or application server's API functions for retrieving the specified server variable - that is it reads from the local server environment.

Next the entry point base64 encodes the retrieved data and digitally signs it. Since the context is still handling a CAM exception as part of an authentication conversation, the entry point now will re-send the original request to the authentication provider after attaching the encoded and signed value in the HTML `<form>` parameter CAM_SecurityBlob.

The authentication provider code looks at every request independently so now when a new request arrives, the same code as for the previous request gets executed. Again the authentication provider will request the trusted environment variable. This time however, the parameter
CAM_SecurityBlob will be present and the attached SecurityBlob parameter's signature will be verified. If the signature verification fails, the request is deemed to have been altered and its trustworthiness cannot be guaranteed. SSO authentication fails at this point. If the signature verification is successful, the decoded parameter contents for the requested sever variable will be returned to the authentication provider. If the variable proves to be empty or not present, the SSO authentication will fail at this point. The authentication provider code can either fall back to basic authentication now or error out. If the value received from the entry point was good though the authentication provider can continue to process the request by entering the next phase.

The preceding paragraphs describe the technical details of an authentication conversation. The digital signature ensures the trustworthiness of the information. This is how IBM Cognos provides trusted environment variables which technically map to server variables.

**Conclusions**

To summarize the support for SSO tokens by IBM Cognos BI,

- Every IBM Cognos BI entry point can provide cookies, HTTP headers and server variables to IBM Cognos BI authentication providers to be used for SSO.
- Depending on the entry point type the set of available server variables may vary. Web server deployed Gateways can supply all SSO token types. Servlet Gateway and Dispatcher should be used for cookies or protected server variables only because there are no simple server variables on Java application servers, only meta-variables.
- Server variables can be retrieved and transmitted internally by IBM Cognos BI in a secure and trustworthy implementation.
- Due to their design, meta-variables should not be used by any SSO configuration. Meta-variable's design intend is to explicitly provide request headers to the server environment, so information passed to the server is represented in the server's environment deliberately.
- Protected server variables like REMOTE_USER and USER_PRINCIPAL are the preferred candidates for SSO tokens. The security of an SSO integration depends on the SSO token type being configured by the administrator. If no support for a safe token (cookie, REMOTE_USER) exists, consider raising an enhancement request with IBM Cognos.

One conclusion we can make is that when a web server deployed Gateway is being used, be aware that it may get circumvented. This means that when accessing a Servlet Gateway or a Dispatcher directly, special attention must be given to the configured SSO token type. In particular, if a meta-variable is being used, protect direct access to your application tier by means outside of IBM Cognos (such as firewalls and IP filtering) to mitigate that risk. IBM Cognos BI does not restrict client access to Dispatchers as a way to prevent someone from sending forged HTTP headers instead of the intended meta-variable. It is important to point out that this is not a flaw in the IBM Cognos BI product but is a result of the actual design of HTTP headers and meta-variables. One example of where using a meta-variable for SSO integration exposes a potential security issue is the configuration for SSO integration with IBM Tivoli WebSEAL based on the HTTP_IV_USER meta-variable. For this kind of setup, either the circumvention of WebSEAL must be prevented or the SSO should be re-designed to use LTPA cookies for SSO to an IBM WebSphere deployed Servlet Gateway.
Another direct implication from the IBM Cognos BI implementation of handling SSO tokens is, that with the ability to prompt the system for trusted environment variables, the trusted sign-on authentication scheme mentioned earlier in the Concepts section becomes more intuitive. Using untrusted SSO tokens bears a great risk since there is no re-authentication during trusted sign-on. If the Cognos authentication providers would use simple environment variables, it would be very easy to trick the system into trusting a forged identity. Therefore all Cognos providers explicitly request trusted environment variables for SSO, mitigating the risk of tampering with server variable values on the way from the entry point to the authentication provider. However, this is no guarantee for secure SSO solutions.

Consider the IBM Cognos BI LDAP authentication provider which employs the trusted sign-on scheme for SSO. If the provider were to be configured to use a meta-variable in its external identity mapping string, Cognos would trust some possibly forged information because although the server variable was securely passed to the authentication provider “as is” from the entry point, a meta-variable is still only a reflection of a request header which can be forged before it even reaches the entry point. This is why the best practice is to only use protected server variables such as REMOTE_USER or USER_PRINCIPAL or cookies for SSO tokens involved with the trusted signon scheme. As will be discussed in detail in the next section, the trusted sign-on SSO scheme is the predominant scheme in IBM Cognos BI as it is used by all Cognos authentication providers except the Active Directory provider in Kerberos mode (default) and the SAP provider.

To support the importance of the trusted sign-on SSO scheme for IBM Cognos BI, the Trusted Signon Provider (TSP) was introduced which further extends the scope of SSO setups. As with an entry point, a TSP is considered to be “part of the system“, meaning it can act as a trusted source when it comes to providing trusted environment variables. In fact, inside a TSP one can attach a trusted environment variable to the request which is subsequently passed to a secondary authentication provider. Technically this will add the signed and encoded HTML \(<form>\) parameter CAM_SecurityBlob just as an entry point would. The secondary provider, which requests a trusted environment variable, receives a valid result without actually triggering a SystemRecoverable exception because the required CAM_SecurityBlob parameter is already available. In addition, since the request is simply passed from the TSP to the full provider, the request doesn't even travel the network. This is why TSPs are named trusted providers in the first place. Their predominant use is to transform arbitrary SSO tokens to REMOTE_USER which to the consuming secondary authentication provider would appear as if the value was read from the protected server variable at the actual entry point. The level of trustworthiness is the same. However the potential of a TSP is much bigger. TSPs are not limited to trusted variables, they can add SDK credentials to a request or even cookies. While the use of cookies may seem a bit obsolete, there may be cookies which require some sort of special handling or perhaps another prompt needs to be added to gather user or system information into the process (for example, to implement two-factor authentication). Therefore the presence of a TSP does not automatically mean it is simply a trusted sign-on - it could be doing other, more complex processing.

**SSO support by authentication provider**

This section provides details about the support for SSO by the out-of-the-box IBM Cognos BI authentication providers.
**LDAP authentication provider**

The IBM Cognos BI LDAP authentication provider supports SSO by trusted sign-on in the form of identity mapping based on trusted environment variables. There is no support for cookies but the trusted environment variable to be used is configurable.

In the LDAP Namespace configuration there is a property called **External Identity Mapping**. This property takes on a string which must syntactically represent an absolute Distinguished Name (DN) or a valid LDAP V3 compliant search filter as defined by RFC 2254. The property supports two IBM Cognos BI configuration macros,

- \${environment(<variable_name>)}
  This macro will be replaced with value of the specified environment variable. There can be multiple occurrences of this macro however they must all request the same variable (see example below). Requesting more than one environment variable is not supported.
- \${replace(<s_source>,<s_pattern>,<s_replaceby>)}
  Replaces all occurrences of \(<s_pattern>\) in \(<s_source>\) with \(<s_replaceby>\). This macro does not support nesting but multiple instances may be present in a single string. Any occurrence of \(\backslash\) (back-slash) or \" (double-quote) in one of the parameter strings must be escaped by the \(\backslash\) (back-slash).

If SSO is enabled by setting the **Use External Identity** property to **true**, the LDAP provider's SSO authentication process works as follows:

- Once a request is passed to the LDAP provider it will be investigated for the response of a SystemRecoverable exception. If found authentication skips the next step.
- If there's insufficient logon data, the provider checks the configuration and if SSO is enabled it will extract the External Identity Mapping string from the configuration. If the \${environment()}) macro is present, the requested environment variable name is retrieved and a SystemRecoverable exception is thrown implicitly by calling the function to retrieve the variable in a trusted fashion.
- Once a request bearing the result of a SystemRecoverable exception is received the retrieved value is considered. If the value is NULL, the environment variable could not be retrieved or was unset. This will cause the SSO to fail and the provider continues by falling back to normal interactive authentication by throwing a UserRecoverable exception asking for credentials. If the environment variable value was retrieved successfully the External Identity Mapping string is transformed using all defined macros. The result of the transformation must be an absolute DN or a valid LDAP search filter.
- Now the provider will bind to the LDAP server using the configured binding credentials (or Anonymous if left empty) and issue a look-up query to identify a user DN. For an LDAP filter, an LDAP search using the configured Base Distinguished Name (BaseDN), the External Identity Mapping string concatenated with some additional filter conditions using sub-tree scope will be issued. For an absolute DN a direct look-up query is sent. Note that the LDAP provider does not support following LDAP referrals and thus the search is limited to the contents of the LDAP server bound to.
- If the result of the look-up query is exactly one entry, the provider assumes that the identity received externally was successfully mapped to one in the LDAP server and continues on. If
there is no result or more than one result, the SSO fails and the provider continues by falling back to normal interactive authentication by throwing a UserRecoverable exception asking for credentials.

- After the successful look-up query, the provider will use it’s Namespace configuration to query all configured attributes from the LDAP and eventually issue a visa for the Namespace. The SSO has succeeded.

It’s important to realize that this SSO is based on mapping strings only. For example, one could pass in a username from Active Directory which is then mapped to an LDAP user of the same name. Whether this is the same actual user is not clear, nor is there any re-authentication of the user identity being passed in. Drawbacks notwithstanding, mapping strings offers flexibility and allows for SSO scenarios where there is no direct support for a specific authentication source in IBM Cognos BI by duplicating users in an LDAP.

The difficulty in getting LDAP provider based SSO to work is usually around crafting the External Identity Mapping string. The use of the macros and knowledge about LDAP filter syntax is essential.

Examples:

- (sAMAccountName=${replace(${environment("REMOTE_USER")}, "EUROPE\", ","))}
  Allows SSO for users from the Active Directory domain EUROPE authenticated by a Microsoft IIS web server to IBM Cognos BI using the LDAP authentication provider accessing the EUROPE domain controller using LDAP. IIS will populate REMOTE_USER with domainuser (ie: EUROPE\montagg). The macro will cut out the domain portion including the back-slash from REMOTE_USER, leaving only the user name (ie: montagg). The user name is mapped to the sAMAccountName property of a user entry in Active Directory.

- (|(uid=${environment("REMOTE_USER")})(cn=${environment("REMOTE_USER")}))
  Allows SSO from any security to IBM Cognos BI through the LDAP authentication provider by mapping the user name passed in REMOTE_USER to either the uid or the cn attribute of a user entry in the accessed LDAP server. The only thing required for this to work is that the security back end populating REMOTE_USER puts in a string which matches an attribute value in LDAP for that user. This is mapping an identity provided externally to some LDAP user. An example of this is SSO from web server authenticated users, regardless of how the web server actually authenticated them.

- (&(uid=${environment("REMOTE_USER")})(departmentNo=17))
  Allows SSO from any security to IBM Cognos BI through the LDAP authentication provider by mapping the user name passed in REMOTE_USER to the uid attribute in the accessed LDAP server. However, the access is restricted to LDAP users which have value of 17 in the departmentNo attribute. This example demonstrates the use of concatenated filters.

As a reminder, the best practice is to use REMOTE_USER or USER_PRINCIPAL for variable names only. Any other variable can possibly be forged, depending on the IBM Cognos BI entry point exposure.
For more information about configuring an LDAP Namespace, refer to the **Configuring IBM Cognos Components to use LDAP** section in the *IBM Cognos BI Installation and Configuration Guide*.

### Active Directory authentication provider

The Active Directory provider supports two different modes of SSO.

The first mode is SSO based on Microsoft Kerberos by leveraging Microsoft APIs to implement a Kerberos handshake between Microsoft Internet Explorer clients, a Microsoft IIS web server, IBM Cognos BI and Microsoft Active Directory (AD).

This mode is only supported if Microsoft IIS is used for a web server to host the IBM Cognos BI Gateway. The Active Directory authentication provider is only available when the IBM Cognos BI Content Manager component is installed on Microsoft Windows. One can separate the IBM Cognos BI application tier and deploy to any supported platform though. However, the IBM Cognos Gateway and Content Manager must be deployed to the same version of Windows (either 32-bit, 64-bit or mix of both) for this SSO mode to work.

The second SSO method is implemented using identity mapping which is very similar to what the LDAP provider implements. The LDAP provider binds to a single LDAP server and issues a user look-up search. However, Microsoft's Active Directory infrastructure can consist of multiple AD servers for multiple, hierarchically structured domain trees, which are organized in what is called a **domain forest**. Since the IBM Cognos BI Active Directory provider leverages Microsoft's APIs directly, it can support this complex multi-server infrastructure and thus access multiple servers when looking up a user where the LDAP provider was limited due to the lack of support for LDAP referrals. However, the AD provider does not provide any configuration options for the look-up search but instead uses the standards defined by Microsoft. The concept remains the same as for the LDAP provider but the variable used for passing in the user identity string is fixed to **REMOTE_USER**, the look-up search compares the string to the sAMAccountName property of an AD user and instead of a single server, the provider can potentially search a single domain, a domain tree or the complete domain forest.

By default, if the **singleSignonOption** advanced parameter is not explicitly specified in IBM Cognos Configuration, the AD provider assumes Kerberos mode for SSO. Only when the singleSignonOption parameter is set to **identityMapping** will the provider refrain from using Kerberos and instead switch to use the identity mapping mode. Changes require a restart of IBM Cognos BI but will not affect any authorization defined so far using the previous setting - it's only affecting authentication.

For Kerberos mode SSO, the IBM Cognos BI authentication provider for Microsoft Active Directory will employ Microsoft's Security Support Provider Interface API (SSPI) to first delegate the user's identity, after being authenticated to the Microsoft IIS server by Kerberos, to the provider. Technically this happens by at least three SystemRecoverable exceptions which transport Kerberos token contents back and forth. The SSO token therefore is the Kerberos token. Once the delegation succeeds, the provider will impersonate the user and attempt authentication to
AD. If successful, the provider will have obtained a security token which is sufficient to have
an authenticated session with AD to retrieve further information about the user and their group
memberships. It's important to point out that in this mode the IBM Cognos BI authentication
provider for Microsoft Active Directory is only the conveyor of messages. The Kerberos token is
never altered or processed by the provider itself. If errors occur, they are the result of the Microsoft
SSPI API calls returning errors.

The delegation process can make the Kerberos SSO mode quite challenging to set up since
many of the prerequisites in the Microsoft infrastructure must be met. However, none of them are
specifically required for IBM Cognos and the vast majority of issues in getting this SSO to work are
solved by taking action on the documented prerequisites for Microsoft Windows and setting up IIS.
The Resources section contains a link to information on setting up IIS for use in an IBM Cognos BI
environment.

In identity mapping mode, the provider will consume the standard environment variable
REMOTE_USER (and will throw a SystemRecoverable to obtain it if necessary) which is expected
to be in one of four possible formats,

- **Username (sAMAccountName)**
  The user’s Windows login name, which in AD is stored in an attribute called
  sAMAccountName. Be aware that sAMAccountName is only unique in a single domain and
  is not applicable in multi-domain environments. An example of a username is montagg.

- **Username prefixed with domain (domain\sAMAccountName)**
  The user’s Windows login name, which in AD is stored in an attribute called sAMAccountName,
  prefixed by the name of the Windows domain that user originates from and a separating
  backslash. By prefixing the domain the name is guaranteed to be uniquely identified in the
  forest. An example is US\montagg.

- **A distinguished name**
  An absolute distinguished name to the user in the AD structure for that domain. For example,
  a DN could be cn=Guy Montag,cn=Users,dc=US,dc=CORP.

- **A user principal name (UPN)**
  A user principal name (UPN) as defined for the user in their Windows profile. An example
  UPN is montagg@US.CORP.

The best practice is to use the username prefixed with the domain name. The reason for this is
that in Microsoft Windows, user IDs are unique only within a single domain. Without the domain
prefix a similar user in a different domain could be found and hinder authentication. The other three
formats will get converted to the domain\sAMAccountName format anyway but the provider needs
to make an additional call to a Microsoft API which can yield errors if the conversion fails. In the
case of an unqualified (ie: no domain prefix) username being provided in REMOTE_USER, the
domain is assumed to be the one the provider is configured to attached to.

The retrieved value of REMOTE_USER is passed to Microsoft APIs unchanged to run
authentication to AD. If the authentication to AD succeeds, the user is authenticated to IBM
Cognos BI and the visa will be issued for the Namespace.
To re-emphasize, identity mapping is based on string values only, the user name as passed in REMOTE_USER can originate from any security layer. If there is a user in AD which has a sAMAccount attribute value which matches the passed string, the SSO will succeed.

Refer to the Configuring IBM Cognos Components to Use an Active Directory Provider section of the IBM Cognos BI Installation and Configuration Guide for details about configuring the Active Directory provider.

**SAP authentication provider**

The IBM Cognos BI authentication provider for SAP leverages the SAP provided Business Application Interface (BAPI). It only supports SSO based on the SAP MYSAPSSO cookie. This cookie can only be created by other SAP software products such as SAP NetWeaver.

If the provider finds the MYSAPSSO cookie, it will be passed “as is” to the BAPI which then attempts to authenticate to SAP. If SAP acknowledges the cookie and the contained SSO ticket to be valid, the user is considered authenticated and the BAPI will proceed to retrieve user information and security group memberships.

**Series 7 authentication provider**

The IBM Cognos BI Series 7 authentication provider is available to support IBM Cognos legacy products. The provider supports SSO based on,

- Trusted environment variables
- Unencrypted clear-text cookies
- A proprietary IBM Cognos Series 7 cookie called AS_TICKET
- A trusted sign-on plug-in developed using the IBM Cognos Series 7 SDK. These plug-ins can consume tokens and provide a value in REMOTE_USER, which is then used to attempt trusted sign-on.

With the IBM Cognos Series 7 Access Manager administration tool one can define what is called an **OSSignon** for every user. This is a string which is saved as a property of a user.

For SSO, a configurable string property that makes use of macro expressions similar to those of the LDAP provider defines which of two supported SSO tokens is to be used. Either a configurable trusted environment variable or an unencrypted clear-text cookie can be retrieved. Once that string value has been parsed, it is compared to all defined OSSignon objects stored in the Series 7 Namespace. If a single match is found, the corresponding user is authenticated.

Since the name of the trusted environment variable can be specified in the configuration property mentioned above, the provider is not fixed on the use of REMOTE_USER. However, for the reasons described in the earlier section The Cognos handling of SSO tokens, the best practice is to stick to protected server variables. Although clear-text cookies are also supported, REMOTE_USER still offers the best security because it’s easy to forge or inject clear-text cookies into a session.
Due to its proprietary nature and its status as a legacy product, Series 7 security is being phased out and replaced by improved support for integration with other providers in the affected products. Therefore, for new applications, the best practice is to leverage AD or LDAP providers wherever possible.

**RACF authentication provider**

The Resource Access Control Facility (RACF) is a security software layer for IBM System z. IBM Cognos BI's provider for this security system is technically little more than a pre-configured LDAP provider. Support for RACF is achieved by requiring IBM Tivoli Directory Services (TDS) to be installed on top of RACF which makes RACF available through LDAP. Consequently the SSO support for RACF is a subset of what the LDAP provider offers - in particular it's restricted to identity mapping based on the REMOTE_USER environment variable.

**NTLM authentication provider**

NTLM is deprecated as of IBM Cognos BI version 10 and is usually only used for test systems. Nevertheless, the provider exists and supports SSO based on the REMOTE_USER environment variable through identity mapping only.

**CA SiteMinder authentication provider**

The IBM Cognos BI CA SiteMinder authentication provider is actually a TSP. It consumes the proprietary CA SiteMinder cookie named SMSESSION and passes it through to CA SiteMinder APIs for validation. Upon successful validation of the cookie, the API will return the name of a SiteMinder UserDirectory and an LDAP DN in the specified UserDirectory to the provider. Combined these two pieces of information uniquely identify a user for CA SiteMinder. A UserDirectory is SiteMinder's concept of a repository which holds users. Technically this could be an LDAP, an AD or a database. Each UserDirectory gets identified by a name which is configured in SiteMinder. The DN is in X.509 notation and specifies the user in that UserDirectory.

By configuration this information gets mapped to a secondary IBM Cognos authentication provider (usually LDAP or AD) and the user's DN is passed to that secondary provider in REMOTE_USER. It's therefore a trusted sign-on based on a mapped identity obtained from a cookie.

**Implementing SSO to IBM Cognos BI**

**Evaluate the options**

When aiming for SSO to IBM Cognos BI, the proven approach is to first identify what SSO tokens types are available at every hop the request takes on its way from the client to the IBM Cognos BI entry point. This determines whether or not one of the out-of-the-box authentication providers can be used directly. If not, the next step is to evaluate if a relatively small Java programming effort in the form of a TSP to transform an existing SSO token to some other token type which is supported by an out-of-the-box provider is sufficient. If even that evaluation comes out negative, undertaking a significant Java programming project to create a full CJAP is a last resort. The following bullet points provide some guidance in determining the available options to get SSO using the out-of-the-box providers.
• If only an arbitrary HTTP header is available, this rules out all IBM Cognos BI providers except LDAP, which technically could use a meta-variable for SSO. This is discouraged for security reasons. As an ultima ratio a TSP could be coded which implements support for an arbitrary HTTP header by whatever means, however this is the least favourable option due to skill and complexity requirements.

• If the protected server variable REMOTE_USER is being provided by some web server or application server along the request's way, this allows for the LDAP, AD and RACF providers as potential candidates. Technically there are the Series 7 and NTLM providers still in the product but these providers are being phased out and therefore are not ideal candidates for new installs. For application server provided variables, use a Servlet Gateway as a best practice. For web server provided variables, deploy an ISAPI or a MOD gateway if possible. If not possible, then use a CGI gateway but be aware that, due to CGI's design, this can negatively impact performance.

• If a cookie is the only candidate for an SSO token, a TSP will be required since the Series 7 Namespace, which technically does support cookies, is being phased out and should not be used for new setups. A simple example for a cookie consuming TSP can be derived from the samples provided with the IBM Cognos BI Authentication Provider SDK in very short time. This would suffice to provide the cookie's value in REMOTE_USER which is supported for SSO by most providers.

It is strongly recommended that CJAPs only be considered after all other options have been exhausted. In addition to the additional license fees for the IBM Cognos BI SDK, the effort to design and create a full-function CJAP is significant and non-trivial and requires advanced Java programming skills. On the other hand, creating a TSP (which also requires an IBM Cognos BI SDK licence) is usually quite straightforward for a competent Java programmer and should not take a lot of time. Implementing a TSP should always be thoroughly investigated before considering a CJAP.

Ultimately the question is how much effort it takes to achieve a seamless authentication experience, not if one can be achieved. Practical limitations such as time constraints and budget will rule out some of the available options but in general, most difficulties with implementing a seamless authentication experience result from a lack of understanding about the backgrounds of the available options.

**Best practice approach**

After the options have been evaluated, configure an IBM Cognos BI test environment and configure the desired authentication providers in IBM Cognos Configuration. Use the Test functionality to verify the providers are able to access the authentication source. It should be noted that there is no test feature for TSPs or CJAPs.

After the authentication providers have been configured and tested, disable anonymous access in Cognos Configuration and try to authenticate to IBM Cognos BI using interactive mode. Point a browser to the IBM Cognos Connection URL, if prompted select a namespace for authentication and provide explicit credentials at the login screen. This should work to be prepared for the next step.
Since IBM Cognos BI processing only starts when a request is being sent to an entry point, it’s reasonable to simulate those requests without the actual authentication solution in place to get the IBM Cognos BI end set up and configured correctly. This approach is also useful for troubleshooting as it allows one to focus on either the IBM Cognos BI functionality or the security system providing the SSO token. It also helps to provide test cases to IBM Cognos Support which cannot necessarily reproduce any environment in detail, especially if it involves third-party software or custom solutions.

Tools that are useful in creating a simplified environment or for simulating security systems that provide SSO tokens include Fiddler2 or Firebug. Web server configuration items like the SetEnvironment statement for Apache web server may also be useful. Refer to Appendix B for detailed examples. Using one of the approaches described in the examples there or any other applicable technique to create the actual SSO token with a fixed value passing it along in the request to IBM Cognos BI is a valid approach.

If simulating the providing security system is not feasible, the minimum requirement is to have a way to capture the HTTP traffic between the client, the authentication layer and IBM Cognos BI. This will help to narrow down whether the token is passed at all. IBM Cognos BI offers sufficient tracing and logging features to determine whether the token passed is received, consumed successfully and whether the actual authentication works. Refer to the Appendix A for more details.

**State of Support**

With all the possibilities the IBM Cognos BI product offers to integrate authentication and to achieve SSO to it, one should expect to have to work with multiple vendors to resolve SSO issues.

This section tries to provides some guidance in regards to establishing,

- What is known to work
- What was tested
- The support provided by IBM Cognos Support or Services

IBM Cognos provides a detailed list of conformance and supported software environments on the [IBM Cognos Software Environments page](https://www.ibm.com/developerworks/ibm/library/ibm-cognos-software-environments/). Along with the list of supported environments there is information about the various levels of support for a particular environment. Pay particular attention to the definition of the support status. From this, one can derive information about what is known to work or expected to work. At the same time there is a corresponding page called the [Cognos Unsupported Environment Policy](https://www.ibm.com/developerworks/ibm/library/ibm-cognos-unsupported-environment-policy/). This policy applies to everything not explicitly listed on the IBM Cognos Software Environments page.

Taking the above into account, one discovers that there can be some gray areas with regards to troubleshooting 3rd party products. The fact that 3rd party products are listed in the IBM Software Environments does not necessarily imply that IBM Cognos Support will troubleshoot that product. The common practice is, that for listed environments, IBM Cognos Support and Services will actively provide sufficient information through various means to enable successful implementation
of the above integration. In particular, the 3rd party prerequisites for integration will be documented and it is expected that these prerequisites have been established. At the same time, one must be prepared to consult the documentation and/or the support organization for the 3rd party product.

Some examples to mention here because of the frequency of occurrence:

- **Integration of IBM Cognos BI products and IBM Tivoli Access Manager WebSEAL**
  Although WebSEAL is part of another IBM product, there is no formal support for integration with it in IBM Cognos BI.

- **Computer Associates SiteMinder**
  There is a dedicated IBM Cognos BI authentication provider for CA SiteMinder. This provider integrates based on the CA SiteMinder cookie SMSESSION. However the provider supports SiteMinder up to version 6.5 only though. For CA SiteMinder versions beyond version 6.5 there is no formal support. SSO integration can be achieved by leveraging the “SiteMinder light” approach documented in IBM Cognos Technotes and documents. It’s best practice to use REMOTE_USER instead of SM_USER though for the reasons described earlier.

- **Microsoft Kerberos**
  The Kerberos based SSO implemented by the IBM Cognos BI authentication provider for Active Directory is based off several prerequisites to be implemented by Microsoft software components. Very frequently it turns out prerequisites are not met, in particular configuration of Microsoft’s Internet Information Server (IIS) is insufficient or required delegation settings are lacking. IBM Cognos documents the prerequisites and shares information on how to meet them in various Technotes and documents.

**Appendix A – Must-Gathers For Troubleshooting**

Below are several must-gather items which should be collected to allow efficient troubleshooting. You will find that there are no particular items for third party software configuration items or the like. The reason is, that the AAA trace enables to see what is being passed to Cognos which is sufficient for troubleshooting the Cognos side of the SSO.

- **Concise description of the situation at hand**
  - Definition/requirement of SSO. SSO from what security to IBM Cognos BI? Is this about a seamless authentication experience or actually passing a user identity?
  - What is the expected user experience? What are the users accessing and what are they supposed to see? Where do they authenticate, if at all?
  - A clear description of user actions. If possible, support description with video or screen captures.

- **Technical information**
  - Software and hardware involved for both client and server
  - Which clients access which servers?
  - Version numbers
  - Platforms (OS level)
  - IP addresses and host names

- **IBM Cognos BI configuration information**
  - IBM Cognos BI components are installed where and how are they distributed?
The cogstartup.xml file (in the configuration directory under the base install directory) from Cognos installations running Content Manager or/and Gateway. In the case of multiple Content Managers or Gateways provide either one.

• The cmplst.txt file from the base install directory.

AAA trace

IBM Cognos BI offers an XML based logging facility called Indication Processing Facility (IPF) which is basically a wrapper around the well known log4j framework provided by the Apache Foundation. To trace the authentication process, there is a template for a logging configuration available which, when activated, will create sufficient output to troubleshoot authentication and SSO in particular.

To enable an AAA trace, follow the steps outlined in the technote at http://www.ibm.com/support/docview.wss?uid=swg21343375.

Once the AAA trace has been created, either provide the log file to support or try reading through it to see if anything obvious stands out. The log file has a clear structure and looking for the word “authenticate” will reveal the calls to the authentication function (the CallStart operation) and their result (the CallEnd operation).

HTTP traces

Whenever possible, HTTP traces will help to speed up the investigation. And HTTP trace can be taken with a tool such as Fiddler2. HTTP traces will show client actions and let us know such things as if the cookies are being passed as expected and if any re-directs are working.

Appendix B – Simulating security systems to provide an SSO token

IBM Cognos Support is often approached by clients experiencing challenges when configuring SSO between some third party software and IBM Cognos. While the IBM support policies usually provide sufficient guidance, there are edge cases where a client is asked to remove some unsupported software component from the picture but it turns out to be not feasible. For those cases and whenever third party software is involved, a troubleshooting best practice to progress such issues is to simulate the third party software by using supported software products. The following examples may be useful to serve as a starting point to follow through with the approach of simulating the actual unsupported product. Often the root cause of an issue is independent of the actual product but rather concerning some specific rundown of HTTP requests. The examples are provided “as-is” and are meant to provide a starting point for those new to setting up SSO. The examples are not comprehensive and the same results can be also be achieved using different techniques.

Example 1: Using Apache web server to provide REMOTE_USER for SSO token

This example allows for the simulation of some web server based security system which would, by design, lead to REMOTE_USER being populated by some value, usually the user name or some other form of ID. With the Apache web server, there is a simple configuration directive
called `SetEnvironment` which allows us to set the value of an arbitrary environment variable, including REMOTE_USER when accessing some resource on the web server. This is useful when simulating using the IBM Cognos BI CGI Gateway only. For simulations including the MOD Gateway and REMOTE_USER, one must actually enable and configure Apache's authentication feature. This is because MODs are web server extensions which only use a subset of the CGI environment which excludes the protected REMOTE_USER variable. The `SetEnvironment` directive does work for all non-standard variables though, even for MOD Gateways.

**Using SetEnvironment:**

- Stop Apache.
- Edit httpd.conf to add configuration for a CGI or MOD gateway in Apache. Refer to [http://publib.boulder.ibm.com/infocenter/cfpm/v10r1m0/topic/com.ibm.swg.im.cognos.cp_inst_cr_winux.10.1.0.doc/cp_inst_cr_winux_id11149ConfigureApachegateway.html#ConfigureApachegateway](http://publib.boulder.ibm.com/infocenter/cfpm/v10r1m0/topic/com.ibm.swg.im.cognos.cp_inst_cr_winux.10.1.0.doc/cp_inst_cr_winux_id11149ConfigureApachegateway.html#ConfigureApachegateway) for more information.
- Add a `<Location>` directive as follows,

```apache
<Location /{cognos_alias}>
  SetEnvironment REMOTE_USER “montagg”
</Location>
```

This will populate REMOTE_USER with a value of “montagg” whenever a resource from this location is requested. The result is that the REMOTE_USER header will be added to the response sent to the client but even more importantly, the variable gets set in the server environment which means IBM Cognos BI's authentication conversation will be able to fetch it through the SystemRecoverable exception process. This allows one to test most Cognos authentication providers by using the CGI Gateway in Apache. One needs to adjust the alias to the actual virtual directory used to access IBM Cognos.

- Save httpd.conf and restart Apache.
- Enable your favorite HTTP monitoring tool.
- Enable the Cognos AAA trace.
- Run the test.

**Using Apache authentication for MOD Gateway + REMOTE_USER**

Refer to [http://httpd.apache.org/docs/2.2/howto/auth.html](http://httpd.apache.org/docs/2.2/howto/auth.html) for some background information used in this technique:

- Stop Apache.
- Edit httpd.conf to,
  - Add configuration for a CGI or MOD gateway in Apache. Refer to [http://publib.boulder.ibm.com/infocenter/cfpm/v10r1m0/topic/com.ibm.swg.im.cognos.cp_inst_cr_winux.10.1.0.doc/cp_inst_cr_winux_id11149ConfigureApachegateway.html#ConfigureApachegateway](http://publib.boulder.ibm.com/infocenter/cfpm/v10r1m0/topic/com.ibm.swg.im.cognos.cp_inst_cr_winux.10.1.0.doc/cp_inst_cr_winux_id11149ConfigureApachegateway.html#ConfigureApachegateway) for more information.
  - Enable the modules `mod_auth_basic` and `mod_authn_file`.
  - Use the Apache command line tool `htpasswd` to create a user file containing at least one user.
• Add the following inside the `<ScriptAlias>` element created earlier,

```
AuthType basic
AuthName “Apache authentication required”
AuthBasicProvider file
AuthUserFile <absolute path to the User file and filename>
Require valid user
```

This will enable Apache's authentication system using very basic settings. Users are defined in a file and the login prompt will be rendered by the browser. This is effective in proving that a user successfully authenticated by Apache will be passed along in REMOTE_USER. One needs to adjust the alias to the actual virtual directory used to access IBM Cognos.

• Save httpd.conf and restart Apache.
• Enable your favorite HTTP monitoring tool.
• Enable the Cognos AAA trace.
• Run the test.

**Example 2: Using Apache web server to send a cookie update on every request**

This example provides a way to rule out cookie handling issues using the Apache 2.2 web server and IBM Cognos BI.

Most authentication products base their authentication on cookies. If any of the IBM Cognos BI components is unable to properly support cookie based session handling of external products, issues can arise with SSO and authentication in general. The concern is that if an external session cookie is frequently updated, then one of the IBM Cognos BI components might not handle the update properly or could use a cached value instead. A typical symptom is when browser clients are working but other type of clients may exhibit unexpected behavior. Examples include heavyweight clients such as IBM Cognos BI Modeller (formerly known as Framework Manager), IBM Cognos TM1 Architect or IBM Cognos Analysis for Excel and mobile clients using IBM Cognos Mobile.

For these scenarios simulating cookie based session handling that includes frequent cookie updates by an external security layer could be a simple and easy way to narrow down the issue.

The basic idea is to use Apache's **mod_header**. This module allows us to manipulate the request and response headers of a request passing through Apache web server. When this functionality is used to add a cookie which gets updated on every round trip, this is a simple way to simulate an external security layer updating its session.

To achieve this:

• Stop Apache.
• Edit httpd.conf to,
  • Add configuration for a CGI or MOD gateway in Apache. Refer to [http://publib.boulder.ibm.com/infocenter/cfpm/v10r1m0/topic/com.ibm.swg.im.cognos.cp_inst_cr_winux.10.1.0.doc/](http://publib.boulder.ibm.com/infocenter/cfpm/v10r1m0/topic/com.ibm.swg.im.cognos.cp_inst_cr_winux.10.1.0.doc/)
cp_inst_cr_winux_id11149ConfigureApachegateway.html#ConfigureApachegateway for more information.

- Enable mod_header by adding the following the line `LoadModule headers_module modules/mod_headers.so`.
- Add a `<Location>` directive as follows,

```xml
<Location /{cognos_alias}>
  Header always append Set-Cookie "%t; path=/;"
</Location>
```

What this does is that a cookie with a name of “t” is added which’s value is the timestamp of the request in milliseconds, which essentially means it’s always updated. One needs to adjust the alias to the actual virtual directory used to access IBM Cognos.

- Save httpd.conf and restart Apache.
- Enable the Cognos AAA trace.
- Enable your favorite HTTP monitoring tool.
- Run the test.

Essentially one will now see a cookie named “t” passed around with every request. If the cookie value doesn't ever change, then it's possible the client is not processing cookie updates. Again, this is just one simple example of how simulating a behavior using some standard software can help resolve an issue. Apache's mod_header functionality is very handy for this and a link to more information about mod_header can be found below in the Resources section.
Resources

- HTTP Headers
- Cookies
- Server variables for Apache HTTP Server
- Server variables for Microsoft IIS
- Server variables for IBM IHS 7
- Microsoft Active Directory Domains and Forests
- Apache mod_header
- Configure Microsoft Internet Information Services 7.x for IBM Cognos 10
About the author

Business Analytics Proven Practices Team

Business Analytics Proven Practices Team

© Copyright IBM Corporation 2013, 2014
Trademarks
(www.ibm.com/developerworks/ibm/trademarks/)