HEXAGON

HxGN EAM Single Sign-On

Version 12.2.1.2
September 2025

Copyright

U.S. Government Restricted Rights Legend

Intergraph Corporation, Hexagon’s Asset Lifecycle Intelligence Division
305 Intergraph Way
Madison, AL 35758

Documentation

Documentation shall mean, whether in electronic or printed form, User's Guides, Installation Guides, Reference Guides, Administrator's Guides, Customization Guides, Programmer's Guides, Configuration Guides and Help Guides delivered with a particular software product.

Other Documentation

Other Documentation shall mean, whether in electronic or printed form and delivered with software or on Intergraph Smart® Support, SharePoint, box.net, or the Hexagon documentation web site, any documentation related to work processes, workflows, and best practices that is provided by Hexagon as guidance for using a software product.

Terms of Use

a. Use of a software product and Documentation is subject to the End User License Agreement ("EULA") delivered with the software product unless the Licensee has a valid signed license for this software product with Intergraph Corporation, Hexagon’s Asset Lifecycle Intelligence Division (“Hexagon”), a Hexagon Group Company. If the Licensee has a valid signed license for this software product with Hexagon, the valid signed license shall take precedence and govern the use of this software product and Documentation. Subject to the terms contained within the applicable license agreement, Hexagon gives Licensee permission to print a reasonable number of copies of the Documentation as defined in the applicable license agreement and delivered with the software product for Licensee's internal, non-commercial use. The Documentation may not be printed for resale or redistribution.

b. For use of Documentation or Other Documentation where end user does not receive a EULA or does not have a valid license agreement with Hexagon, Hexagon grants the Licensee a non-exclusive license to use the Documentation or Other Documentation for Licensee’s internal non-commercial use. Hexagon gives Licensee permission to print a reasonable number of copies of Other Documentation for Licensee’s internal, non-commercial use. The Other Documentation may not be printed for resale or redistribution. This license contained in this subsection b) may be terminated at any time and for any reason by Hexagon by giving written notice to Licensee.

Disclaimer of Warranties

Except for any express warranties as may be stated in the EULA or separate license or separate terms and conditions, Hexagon disclaims any and all express or implied warranties including, but not limited to the implied warranties of merchantability and fitness for a particular purpose and nothing stated in, or implied by, this document or its contents shall be considered or deemed a modification or amendment of such disclaimer. Hexagon believes the information in this publication is accurate as of its publication date.

The information and the software discussed in this document are subject to change without notice and are subject to applicable technical product descriptions. Hexagon is not responsible for any error that may appear in this document.

The software, Documentation and Other Documentation discussed in this document are furnished under a license and may be used or copied only in accordance with the terms of this license. THE USER OF THE SOFTWARE IS EXPECTED TO MAKE THE FINAL EVALUATION AS TO THE USEFULNESS OF THE SOFTWARE IN HIS OWN ENVIRONMENT.

Hexagon is not responsible for the accuracy of delivered data including, but not limited to, catalog, reference and symbol data. Users should verify for themselves that the data is accurate and suitable for their project work.

Limitation of Damages

IN NO EVENT WILL HEXAGON BE LIABLE FOR ANY DIRECT, INDIRECT, CONSEQUENTIAL INCIDENTAL, SPECIAL, OR PUNITIVE DAMAGES, INCLUDING BUT NOT LIMITED TO, LOSS OF USE OR PRODUCTION, LOSS OF REVENUE OR PROFIT, LOSS OF DATA, OR CLAIMS OF THIRD PARTIES, EVEN IF HEXAGON HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

UNDER NO CIRCUMSTANCES SHALL HEXAGON'S LIABILITY EXCEED THE AMOUNT THAT HEXAGON HAS BEEN PAID BY LICENSEE UNDER THIS AGREEMENT AT THE TIME THE CLAIM IS MADE. EXCEPT WHERE PROHIBITED BY APPLICABLE LAW, NO CLAIM, REGARDLESS OF FORM, ARISING OUT OF OR IN CONNECTION WITH THE SUBJECT MATTER OF THIS DOCUMENT MAY BE BROUGHT BY LICENSEE MORE THAN TWO (2) YEARS AFTER THE EVENT GIVING RISE TO THE CAUSE OF ACTION HAS OCCURRED.

IF UNDER THE LAW RULED APPLICABLE ANY PART OF THIS SECTION IS INVALID, THEN HEXAGON LIMITS ITS LIABILITY TO THE MAXIMUM EXTENT ALLOWED BY SAID LAW.

Export Controls

Intergraph Corporation, Hexagon's Asset Lifecycle Intelligence division ("Hexagon"), and a Hexagon Group Company's software products and any third-party Software Products obtained from Hexagon, its subsidiaries, or distributors (including any Documentation, Other Documentation or technical data related to these products) are subject to the export control laws and regulations of the United States. Diversion contrary to U.S. law is prohibited. These Software Products, and the direct product thereof, must not be exported or re-exported, directly or indirectly (including via remote access) under the following circumstances:

a. To Cuba, Iran, North Korea, Syria or the Crimean, "Donetsk People's Republic", "Luhansk People's Republic", or Sevastopol regions of Ukraine, or any national of these countries.

b. To any person or entity listed on any U.S. government denial list, including but not limited to, the U.S. Department of Commerce Denied Persons, Entities, and Unverified Lists, http://www.bis.doc.gov/complianceandenforcement/liststocheck.htm, the U.S. Department of Treasury Specially Designated Nationals List. Visit www.export.gov for more information or follow this link for the screening tool: https://legacy.export.gov/csl-search.

c. To any entity when Licensee knows, or has reason to know, the end use of the Software Product is related to the design, development, production, or use of missiles, chemical, biological, or nuclear weapons, or other un-safeguarded or sensitive nuclear uses.

d. To any entity when Licensee knows, or has reason to know, that an illegal reshipment will take place.

Any questions regarding export/re-export of relevant Hexagon software product, customized software, Technical Data and/or third-party software obtained from Hexagon, its subsidiaries or distributors, should be addressed to Hexagon's Export Compliance Department, 305 Intergraph Way, Madison, Alabama 35758 USA or at exportcompliance@intergraph.com. Customer shall hold harmless and indemnify Hexagon and a Hexagon Group Company for any causes of action, claims, costs, expenses and/or damages resulting to Hexagon or a Hexagon Group Company from a breach by Customer.

Trademarks

Intergraph®, the Intergraph logo®, Intergraph Smart®, SmartPlant®, SmartMarine, SmartSketch®, SmartPlant Cloud®, PDS®, FrameWorks®, I-Route, I-Export, ISOGEN®, SPOOLGEN, SupportManager®, SupportModeler®, SAPPHIRE®, TANK, PV Elite®, CADWorx®, CADWorx DraftPro®, GTSTRUDL®, CAESAR II®, and HxGN SDx® are trademarks or registered trademarks of Intergraph Corporation or its affiliates, parents, subsidiaries. Hexagon and the Hexagon logo are registered trademarks of Hexagon AB or its subsidiaries. Other brands and product names are trademarks of their respective owners.

The following are trademarks and service marks owned by Intergraph Corporation d/b/a Hexagon or its subsidiaries worldwide: hexagonppm.com/trademarks.

About this brief

This brief provides configuration and customization information for the use of Single Sign-On with HxGN EAM. This document will detail the steps necessary to configure HxGN EAM or SAML message authentication with ADFS (3.0 or later), Azure AD, or Okta. This document also provides customization options.

This brief is intended to supplement the documentation of this feature. It is not comprehensive and may not include all the details about this functionality.

Attachments

This document contains attachments. The attached files include:

- claimrules.ps1
- rp.properties

To access these files, browse to the left-hand menu in the browser for this document, and then click **2 attachments**. The attachments view opens with the selected documents available for downloading.

Overview

The HxGN EAM Single Sign-On module authenticates users by exchanging SAML messages with a SAML-aware Single Sign-On server known as an Identity Provider (IDP).

The module within an application responsible for communicating with the IDP is called a Service Provider (SP).

A typical HxGN EAM installation will be configured with one SP and one IDP. Most of this document will focus on the configuration necessary for this deployment scenario. It is possible to configure multiple SP's and/or multiple IDP's. See “Multiple identity providers” and “Multiple service providers.”

The instructions assume both HxGN EAM and the IDP have already been installed, presumably on two different machines.

SAML authentication requires that a trust relationship be established first between the IDP and SP servers. An IDP will only allow an SP to use its authentication service if it has been configured to recognize the SP. The SP, in turn, will trust the SAML claims it receives from the IDP because it has been provided with the digital certificate the IDP uses to sign SAML messages. The trust relationship between IDP and SP is established by exchanging metadata files between the two applications. The IDP and SP each generate a metadata file, an xml file that is passed to the other party with information necessary to create the trust relationship.

Single Sign-On is supported through an external logon feature. When HxGN EAM is configured for external logons, it bypasses its own logon screen and uses an external server to authenticate users. The main reason for employing an external server is to support single sign-on. By default, the external logon mechanism makes use of EAM's Single Sign-on module. Alternatively, a custom logon module may be substituted to connect to a wide variety of external SSO servers.

See "Customizing a Single Sign-On logon module".

This document discusses SSO configuration for both cloud and on-premise installations of EAM. The sections describing how to configure EAM files and directories only apply to on premise installations.

Logon types

HxGN EAM supports the following strategies for authenticating a user during the logon process:

- **Standard Logon:** HxGN EAM prompts for a username and password, which are checked against the database.
- **LDAP logon:** HxGN EAM prompts for a username and password. These credentials are used to connect to an LDAP server to authenticate the user.
- **External logon:** HxGN EAM does not prompt for username/password. Instead, the user is redirected to an external server which is responsible for authentication.
- **Resource Owner Password Credentials Grant (ROPC) logon:** This mode of logon is only supported for SOAP/REST Web Services in HxGN EAM. In this mode, the authentication is carried out using the ROPC flow of OpenID Connect.
- **OpenID Connect Authorization Code Grant:** This mode of login is only supported for SOAP/REST Web Services in HxGN EAM. The details about this authentication are presented in the HxGN EAM OpenID Connect technical brief.
- **Internal Authentication Mode:** In this mode of login, the credentials are validated against the database irrespective of the value of the LGNEAM or LGNCON install parameters. This mode of login is supported for Web login and SOAP/REST Web Services in HxGN EAM.

These logon types may be configured for each tenant, and for each product within a given tenant. Different tenant/product combinations may freely mix the logon types in any desired combination.

Internal authentication mode

In this mode of login, the credentials (username and password) are validated against the database irrespective of the value of the **LGN*** install parameters.

The details about this authentication mode for SOAP/REST Web Services in HxGN EAM are presented in the HxGN EAM OpenID Connect technical brief.

To use the Internal Authentication Mode for Web login, the HxGN EAM login URL should be appended with a query string parameter authenticationmode with a value of **internal**. By accessing the HxGN EAM URL with this additional parameter, the user will be prompted for username and

password which will be validated against the database irrespective of the value of the install parameter, **LGNCON**.

Sample

<protocol>://fully_qualified_eam_server_name:port/web/base/logindisp?tenant=<EAM_TENANT>&authenticationmode=internal

Install Parameters for Single Sign-On

Install parameters for logon configuration

The four logon strategies (standard, LDAP, external and ROPC) are configured on a per-product basis. These are the install parameters for product types:

- LGNCON: web service logons
- LGNDBR: Databridge logons
- LGNMOB: Mobile logons
- LGNCAL: Caller and portal logons
- LGNEAM: Browser-based logons

Each install parameter can be set to any of the following possible values:

- STD: standard logon
- LDAP: LDAP logon
- EXTERN: external logon server
- ROPC: external logon server using the OpenID Connect Resource Owner Password Grant authentication flow. This option is not supported LGEAM logons

  Note: See the HxGN EAM OpenID Connect technical brief for configuring ROPC.

Prerequisites for Single Sign-on configuration

EAM should be installed, with SSL enabled. The procedure for enabling SSL is discussed in other EAM documentation.

In one of the yaml configuration files (under depconfig/properties/external), set core.ssl to true. This setting should be included in the file containing other settings for the ‘core’ block

Ensure that core.server is set to the public name of the EAM server

SSO Configuration screen

Some configuration may be performed on the 'SSO Configuration' screen rather than in the yaml. In particular, the ws-trust settings may be configured either in the yaml or on the screen. Configuring ws-trust on the screen makes it possible to have different settings for different tenants. The various ws-trust settings are discussed in the SSOLogonDef section below.

Note: The SAML claims used by ws-trust are always taken from the yaml; the SAML claims on the SSO Configuration screen are only used for OIDC.

This screen may also be used, (cloud only) to upload the IDP's federation metadata file to EAM, and to download the cloud server's metadata file (sp.xml). The latter is used to configure the IDP (relying party or application).

Configuration of the IDP

ADFS

- Create a relying party using the EAM sp.xml file (available either from the SSO Configuration screen (cloud) or in ssoclienthome (on premise))
- Use the EAM powershell script (claimrules.ps1) to configure the claim rules

Azure AD

1. Create a new application:
    - Open "Enterprise Applications" and select "New Application"
    - Enter "Azure AD Saml Toolkit" in the search box and select the Azure AD Saml Toolkit application

2. SAML Configuration:
    - In the new application, open "2. Set up single sign on" and then select "SAML" as the single sign-on method
    - "Basic SAML Configuration" section
        - Select "Upload metadata file" and upload the metadata file (sp.xml) provided by EAM. For cloud deployments, sp.xml is obtained on the SSO Configuration screen. For on premise deployments, sp.xml will reside in ssoclienthome once basic SAML setup is completed in the yaml
        - On the 'Basic SAML Configuration tab:

        - Set the Sign on Url to the AssertionConsumerService[Location] property in sp.xml
        - Save the information
    - “Attributes and Claims” section: configuration of SAML claims is described in the next section

3. Assign users to the application.

Okta

Okta tutorial: https://help.okta.com/en-us/Content/Topics/Apps/Apps_App_Integration_Wizard_SAML.htm#Task

See the next section to configure the SAML claims

SAML Claims

The SAML claims defined in the RP (ADFS Relying Party or Azure AD application) must match the SAML claims defined in EAM. If the RP defines SAML claims that EAM does not use, the extra claims are harmless but will be ignored.

For ADFS these claims can be created using the claimrules.ps1 script.

Claim Name	Value with Azure AD	Value with Okta
http://schemas.hexagon.com/claims/Identity	user.userprincipalname	user.login
http://schemas.hexagon.com/claims/Description	user.displayname	
http://schemas.hexagon.com/claims/DisplayUser	user.userprincipalname	user.login
http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress	user.mail	user.email
http://schemas.microsoft.com/ws/2008/06/identity/claims/role	A constant, or one of more group names	Group attribute statement

http://schemas.hexagon.com/claims/Customer	Constant string	Constant string
http://schemas.xmlsoap.org/ws/2005/05/identity/claims/upn	user.userprincipalname	user.login
http://schemas.xmlsoap.org/claims/Group	user.assignedroles	Group attribute statement

Notes on these claims:

- Identity: The value of this claim is an identifier that uniquely identifies the end user in the EAM database (r5users table). It must correspond to either the usr_code or the usr_externcode in r5users. The recommended value for this claim is the UPN. However, if externcodes are already set up in r5users with a different value (e.g. the email address), that value should be used instead.
- For on premise installations, the name identifier type must be set in 'transient' in the IDP
- Role: the role claim is used only to perform Just In Time (JIT) user creation in EAM. The simplest way to configure JIT user creation is to (1) return a list of the groups the user belongs to in the role claim, (2) create corresponding roles in EAM, and (3) set the SSOROLES install parameter. This install parameter specifies the order of priority of roles in the SAML response when more that one value is returned.

Create idp.xml

1. Open this location with a browser and save the file: https://<adfsserver:port>/FederationMetadata/2007-06/FederationMetadata.xml
2. Rename the file to idp.xml.
3. Use some editor to add line breaks
4. Delete everything from ds:Signature (the second line of the file) up to but not including <IDPSSODescriptor
5. The first two lines of the resulting file should look like
   - <EntityDescriptor
   - <IDPSSODescriptor
6. Be sure to save the file as plain ascii
7. Copy idp.xml to overrides/sso/.

Configuring external logons

Create a yaml file for the sso configuration:

- File location: depconfig/properties/external
- File name: the file should be called ‘n_somestring.yml’, where:
  - n is a number from 1-9 which is unique in that directory
  - ‘somestring’ can be any descriptive string
  - The file must have a ‘.yml’ extension
- Editing the file
  - The file must be a plain text file.
  - The file should initially contain the entire ‘sso’ block from properties\internal\default.yml.
  - Please note that no files in the internal directory may be edited directly.

The file contains three main sections:

- ssoLogonDef
- customLogonDef
- tenant

The sections must occur in the above order. At a minimum there must be one logon definition section (either ssoLogonDef or customLogonDef) and one tenant section.

Typically, this file will contain only one logon definition. If there is a need to connect to different external servers for different tenants and/or products, multiple logon definitions may be provided. The file may contain both ssoLogonDef and customLogonDef sections, and each section may be repeated as many times as is necessary.

Note: Values in this file are case sensitive.

Setting up the external logon definition file

1 Set sso.enabled to true

2 Set sso.idpentityid to the entity id in idp.xml

3 Set sso.eamentityid to a valid value. The value must be globally unique, so a good choice would the server and port EAM is running on (https://fully_qualified_eam_server_name:port). If the port EAM is running on changes, the entity id does NOT need to change. The entity id is just an identifier, not a URL

SSOLogonDef section

The ssoLogonDef block is used only when connecting to a SAML2-compliant SSO server. It must not be used for custom SSO implementations.

- **name**
  Each logon definition must have a unique name. The name must be unique across all ssoLogonDef and customLogonDef blocks.
- **logonURL**
  The URL of the HxGN EAM module that communicates with the SAML server.
  - Since the browser will be redirected to this URL, the URL must be accessible from the browser.
  - The default value will normally not need to be changed.
- **logoutURL**
  The URL to which the browser will be redirected when the user logs out of HxGN EAM.
  - **Note:** Do not set the logoutURL to the HxGN EAM login page. If the logoutURL points to the login page, users will be unable to log out at all.
- **ssoLogoutURL**
  The URL to which the browser will be redirected if the HxGN EAM session times out.
- **STSEndpoint**
  - Use a browser to access the ADFS mex address endpoint (see 'STSMexAddress' below)
  - In the resulting xml document search for an address ending in "2005/usernamemixed"
  - Use the entire URL for the STSEndpoint
- **STSKeyType**
  - The default value should not be altered.
- **STSMexAddress**

  To find the Message Exchange Address using the ADFS console:
  - Navigate to Service → Endpoints.
  - Locate the Metadata section.
  - The Metadata section should contain an entry whose type is WS-MEX.
  - Copy the URL Path for this entry.
  - The complete mex address is a URL using this path. The mex address will look something like https://<adfsserver>:<optional port>/adfs/services/trust/mex.
- **STSPolicyID**
  Should be left blank when connecting to ADFS.
- **userAttribute**
  The name of the SAML attribute containing the userid (the unique identifier for the user).
  - This value must exactly match, including case, a Claim Type in ADFS
    - See the Claim Descriptions screen (under the Service tab)
  - Filter
    - In the event that the userid is a substring of the value returned in the SAML claim, a filter may be specified to extract the userid from the SAML value.
    - Requirements for the filter string:

        - It must be a valid Java regular expression
        - Group number 1 will be used to extract the userid substring. If other groups exist they must be non-capturing.
    - If the filter is omitted the entire SAML value will be used.
- **primaryRoleAttribute**
  The name of the SAML attribute containing an EAM role name
        - The value must match a role defined on the EAM roles screen.
        - This value must exactly match, including case, a Claim Type in ADFS
        - See the Claim Descriptions screen (under the Service tab)
        - If this value is omitted, user records must already exist in the EAM database
        - Filter (see details under the userAttribute above)

- **identityProviderID**
  A unique identifier for the Identity Provider (e.g. ADFS).
    - By default this value will be automatically populated using the idpEntityID from the yaml configuration.
    - The value only needs to be entered manually if more than one Identity Provider is configured. The ID should be copied from the entityID attribute in the appropriate idp*.xml file.
    -
- **identityProviderType**
  Defaults to ADFS. This setting should be left unchanged when connecting to ADFS.
- **logonRedirectTimeout**
    - When control returns to EAM from an SSO server, parameters are passed to EAM in an encrypted string by means of an http post from the browser.
    - To minimize the possibility of a replay attack, the encrypted string contains a timestamp indicating how long the string remains valid.
    - The timestamp value is given in seconds.
    - The server defaults this value to 300 seconds. The minimum value is 5 seconds; any smaller value will be ignored.
- **faaConfig**
  This section is only used when EAM is running in the SAAS environment.
    - **tenantCookieDomain**
      The domain name to use when setting the FAACustomerID cookie
        - The name will normally begin with a dot (‘.’)
        - If this property is present the tenantID must be of the form “CustomerID_somestring”.
    - **qualifyUserNames**
      A Boolean value, If true, the userid sent in a WS-Trust request will be of the form (CustomerID_userid)
        - The customerID will be extracted from the tenantID passed in the SOAP request.
        - If this attribute is set to true, the tenantID must be of the form “CustomerID_somestring”
    - **transmitTenant**
      Transmit the entire tenant id, not just the customer id portion, with a ws-trust request.

    - The transmitTenant and qualifyUserNames elements should not be used together. If both elements are present, the qualifyUserNames element will be ignored.
- **internalUserAttribute**
  The name of a SAML claim whose value is a displayable user name.
    - If the logon process results in the automatic creation of a record in r5users, the value of this attribute will be used to populate the usr_code column (provided the value is <= 30 characters). This column will be displayed to end users on various screens.
- **emailAttribute**
  The name of a SAML claim whose value contains the user's email address.
    - If the logon process results in the automatic creation of a record in r5users, the value of this column will be used to populate the email address in r5users.
- **userDescriptionAttribute**
    - The name of a SAML claim whose value contains a description of the user. The value will be used to populate the description in r5users.

Tenant section

The tenant section specifies the logon definition(s) used by a particular tenant. A tenant section may be added for each tenant that uses external logons for one or more products.

The default tenant entry:

- A default tenant section may be added with the name set to "__default__".
- This entry is used when no tenant entry is provided for a tenant who uses external logons.
- If all tenants use the same logon definition it is not necessary to add entries for specific tenants
- Entries for individual tenants are only necessary if those tenants use different logon definitions.

Tenant name: the name must match exactly, including case, the tenant name in ewsconfig.xml.

Product section:

- A tenant section will contain one or more product elements
- The product id must be one of these:
    - eam
    - connector
    - databridge
    - mobile
        - The product's logondef must match exactly the name of either an ssoLogonDef or a customLogonDef element.
        - The 'eam' product serves as the default. If all products for a given tenant use the same logon definition, then only the 'eam' product should be specified.
- **eamEntityID**
  A unique identifier for the Service Provider (e.g. EAM).
    - This value must be a valid URL. Any unique URL will suffice.

    - A reasonable choice is the AssertionConsumerService location in the sp.xml file.
- **metaalias**
  The metaAlias of the Service Provider used by this tenant.
    - The value should be copied from the metaAlias attribute in the appropriate sp*-extended.xml file.
    - Note that the metaAlias always includes a leading “/”.

Relationship between logon install parameters and tenant/product configuration in the sso configuration file

The Logon install parameters (LGN*) control the type of logon used for the corresponding product. The tenant/product information in sso configuration file controls the type of external logon used when the install parameter is set to EXTERN. If the install parameter is not set to EXTERN, the sso configuration file is ignored for that tenant/product combination.

Using SSL

For instructions on configuring EAM to use SSL, refer to the HxGN EAM installation guides (Enterprise and Asset Sustainability edition).

EAM must be configured to use SSL. ADFS will not send a SAML response message over an unencrypted connection. EAM may be configured to use SSL only during the logon process or to use SSL for all communication.

To force EAM to use SSL for all communication, set core.ssl to true in the appropriate yaml configuration file. Setting this flag to true has the following consequences:

- The AppServerHost property in MPConfiguration.xml is configured for SSL. Since this property is used by the EAM SSO module when returning from ADFS, the result is that ALL communication with EAM will occur over SSL.
- SSL is enabled for the logonURL property in the sso configuration file.

  Note that whether EAM uses SSL for all communication does not depend on whether the user initially accesses EAM over an SSL connection.

Signed certificates: EAM opens http connections to ADFS and to other modules within EAM. If these connections employ SSL, the certificate returned by the server (ADFS, or the instance of Apache installed with EAM) must be signed by a CA that can be validated using the certificate chains present in the EAM cacerts file (located in <eamhome>/java/jre/lib/security)..

- The certificate returned by ADFS will be validated against cacerts if any of the following EAM functionality is used:
    - Web services, including all connector requests (mobile, databridge, etc.)

    - Electronic signatures
    - Advanced reports
- The certificate returned by ADFS is called “Service Communications” on the ADFS Certificates screen.

Processing flow when external logon is enabled

When a user attempts to log onto HxGN EAM, the browser will be redirected to the external Single Sign-On server.

If the SSO server determines the user is not logged onto the system, it will authenticate the user by some method, typically by prompting for logon credentials.

Once the SSO server has authenticated the user, it will redirect the user back to HxGN EAM.

Referencing external user IDs in EAM

External systems may identify a user by email address, UPN, or some other method. This external identifier may not correspond to the user code in the r5users table. To facilitate the mapping from an external user id to an r5users record, a new column (usr_externcode) has been added to r5users. This new column may be populated from the User Setup screen by means of the **External User ID** field, which is hidden by default.

When a user is authenticated, the user id returned to EAM in the SAML response (in the SAML claim specified by the **userAttribute** element in the sso configuration file) can match either the externcode or the usrcode in r5users.

If the user id in the SAML response does not match an existing r5users record, a new record may be created automatically (assuming proper role configuration).

- **externcode column**
  Set based on the userAttribute SAML claim in sso configuration file
- **usrcode column**
    - Set based on the internalUserAttribute SAML claim in sso configuration file if that value is present and not too long for the column
    - Otherwise, set to the externcode if that value is not too long for the column
    - Otherwise, set to a randomly generated number.

Redirecting the browser back to HxGN EAM

When HxGN EAM redirects a browser to the SSO server, it includes the return URL on the query string.

The general format of the url will be http://eamserver:eamport/web/base/ssoservlet?parameters...

The SSO server must redirect the browser to the ssoservlet target. Redirects to other targets will not work.

It is recommended that the SSO server add a token to the return URL indicating that the request comes from a custom SSO server. Specifically, the modified URL would be http://eamserver:eamport/web/base/ssoservlet/ssocustom?parameters...

Note: The 'web/base" portion of the URL will vary if the base URI has been changed. The redirect must include the parameters passed on the query string to the SSO server.

Closing the browser after drill backs

Every time a user performs a drill back two new browser windows are opened. To close these windows when the user logs out of HxGN EAM, set the logoutURL in the sso configuration file to http://eamserver:eamport/web/base/ssoservlet/closewindow.

Excluding the RequestAuthnContext parameter from the SAML Authentication Request

The SAML Authentication request initiated from EAM has the RequestedAuthnContext parameter enabled by default with the authentication method set to PasswordProtectedTransport.

This could result in SAML Authentication failures in some cases like the 'password-less' or 'windows-hello' authentication supported by Azure AD where the issue could be resolved by excluding the RequestAuthnContext parameter from the SAML Authentication Request.

To exclude the RequestedAuthnContext parameter from the SAML Authentication Request the value of the install parameter, EXRASAML, should be set to YES.

Configuring ADFS with Infor Federation Services

The HxGN EAM Single Sign-On module is designed to work in conjunction with Infor Federation Services (IFS). IFS is a separate application designed to automate the ADFS configuration process and facilitate the management of user accounts and security settings.

For more information on IFS please see the "***Technology Stack: Generic Solution for Infor Federation Services (IFS) Release"*** knowledge base article. Contact Infor Support organization for more information.

The IFS Administration Guide describes how to register an application such as EAM with ADFS. In particular, the *Configure Applications* section describes how to configure an application through IFS. When configuring EAM, use the following instructions instead of the instructions in that section.

FS application configuration utility

1 Open the utility:

- Right-click the IFSConfiguration file.
  - Assuming the default IFS installation directory, this file is located in c:\program files\Infor\Federation Services\bin
- Select Open.

2 Expand the Applications section and enter the following values:

- **Name**
  Enter a descriptive name.
- **Type**
  EAM
- **URL**
  The value for the Location attribute of the AssertionConsumerService element in sp.xml (in <ssoclienthome>)
- **Description**
  Enter a suitable description
- **Relying Party Identifier**
  The value entered for the "entityID" attribute in sp.xml.
- **Encryption Certificate Public Key**
  Not used by EAM.

3 Click **Add** to save the information.

4 Close the utility.

IFS configuration service

1 Open the IFS Configuration Service in a browser

- The URL is https://ifsservername:port/IFS.

2 Navigate to Configure → Applications

- Select the newly created EAM application record.
- Click **Activate.**

3 Navigate to Manage → Master Data

- Select **EAM Template** and click **Details**.
- Click **+**, then enter the name of an EAM role under **node name.**
- Add additional node names/roles as desired.
- Save the data.

4 Navigate to Manage → Users

- For each user who is authorized to access EAM, select the user record and assign the appropriate EAM Template value.
- Save the data and exit.

Validating SAML claims for tenants

ADFS may be configured to return SAML claims containing the names of tenants a given user is authorized to access. One reason to configure ADFS in this fashion is to allow some users to access a production but not a test database, others to access only a training environment, and so on.

The steps required to configure ADFS to support this feature are beyond the scope of this document and will not be discussed here.

To configure EAM to validate SAML claims for tenants:

- ADFS must return a SAML claim such that
  - The name of the SAML claim is http://schemas.infor.com/claims/Tenant
  - The value of the SAML claim is the current tenant name (passed on the query string).
- EAM install parameter VLENVCLM
  - By default, the value of this parameter is set to OFF.
  - To validate SAML claims for tenants set the value to ON

Configuring multiple identity providers and/or service providers

Typically, EAM is configured to use one IDP and one SP. In this case, the configuration files in <ssoclienthome> do not need to be edited manually—the deployment scripts modify these files based on the settings in sso configuration file.

In general, creating multiple IDPs or SPs involves the following changes:

- For each additional IDP, create two new files called idp-<identifier>.xml and idp-<identifier>-extended.xml.
- For each additional SP, create two new files called sp-<identifier>.xml and sp-<identifier>-extended.xml.
  - The identifier may be any desired value; it must be unique across all IDPs and SPs.
- The entityID for each IDP or SP must be added to fedlet.cot
  - The sun-fm-trusted-providers key must contain a comma-separated list of the entity IDs for all SPs and IDPs.

Multiple identity providers

Multiple IDPs may be configured to authenticate different tenants and/or products against different ADFS instances.

The idp.xml file is exported from ADFS and edited as described earlier. The only change to idp-extended.xml is to set the correct entityID.

Multiple service providers

Typically, EAM is configured to use a single service provider. The main reason to configure more than one service provider is if different IDPs are configured to return different SAML claims.

Two configuration files must be created for each service provider in <ssoclienthome>: sp*.xml and sp*-extended.xml.

The ensuing discussion will illustrate how to configure two service providers to control user access to production and test environments.

For the test service provider create files in <ssoclienthome> called sp-test.xml and sp-test-extended.xml.

For the prod service provider create files in <ssoclienthome> called sp-prod.xml and sp-prod-extended.xml.

Updates to the two sp*xml files

1. Set the entityID to a valid value.
    - The requirements for a valid entityid for a service provider are the following:
        - The entityID must be unique across all service providers and identity providers; and
        - The entityID must be in the form of a URL.
    - These are the suggested entity IDs:
        - Production SP: https://<server>:<port>/sso/fedletapplication/prod
        - Test SP: https://<server>:<port>/sso/fedletapplication/test
2. Add the meta alias to the AssertionConsumerService Location
    - A metaalias is a unique identifier for an SP; it is used internally by the SP client packages.
    - Find the Location attribute for the AssertionConsumerService
    - Append this string: "/metaAlias<the actual metaalias>"
    - For example, if the metaalias is '/prod', the modified location would be "…sso/fedletapplication/metaAlias/prod"
    - Note that the term metaAlias must be entered exactly, including case.

Updates to the two sp*-extended.xml files

- Set the entityID to the value entered in the corresponding sp*.xml file
- Set the metaalias to the correct value. Note that the metaalias must begin with a slash ("/")

Updates to the sso configuration file

- Set the eamEntityID and metaalias to appropriate values for the tenant/product entries.

Update fedlet.cot

- The entity IDs for all Identity Providers and Service Providers must be entered in fedlet.cot (in <ssoclienthome>)
- The entity IDs form a comma-separated list on this line: sun-fm-trusted-providers

ADFS configuration: for each sp*.xml file, create a corresponding relying party in ADFS.

Configuring Infor STS with HxGN EAM

Infor Security Token Service (Infor STS) facilitates and provides standards-based single sign-on (SSO) services to users of Infor business applications when federated with an identity provider such as Microsoft Active Directory Federation Service (ADFS).

Note: HxGN EAM does not require special configuration tasks for integrating with Infor STS outside of the standard SSO configuration required for HxGN EAM with ADFS, detailed in the previous sections, except for the details mentioned below. For Infor STS configuration, see the *Infor Operating Service Installation Guide*.

Create idp.xml

1 Open the STS Admin UI in the machine on which Infor OS is installed and navigate to the Settings Menu. Go to the Download Federated Metadata section and download the SAML IdP Metadata XML. https://<sts-server>:<port>/inforsts/rest/metadata/00000000000000000000000000000000/saml/idp

2 Don't make any changes to the file and rename it to idp.xml.

3 Copy the idp.xml to overrides/sso/ folder in the EAM installation directory.

SSOLogonDef section

The following section only describes those properties which are specific to Infor's STS. The initial setup must be done as it is described in the SSOLogonDef Section mentioned earlier in this guide and the specific settings mentioned below must be reconfigured specifically for Infor's STS. These properties may be configured using the SSO Configuration screen rather than in the yaml. configuration files. Configuration via this screen is needed primarily in the SaaS environment. When the screen is used, some values (SAML claim names, EAM entity ID) are still read from the configuration files.

- **STSEndpoint**
  - Use a browser to access the STS mex address endpoint (see 'STSMexAddress' below)
  - In the resulting xml document search for an address ending in "idp/usernamemixed"

    - Use the entire URL for the STSEndpoint
- **STSMexAddress**

  To find the Message Exchange Address using the STS Admin UI:
    - Navigate to Settings Menu → Download Federated Metadata.
    - Locate the Download SAML/WS-Fed IdP Metadata XML link.
    - The Metadata section under the SecurityTokenServiceEndpoint section should contain an Address entry.
    - Copy the URL Path for this entry.
    - The complete mex address is a URL using this path. The mex address will look something like https://<sts-server>:<port>/inforsts/rest/metadata/00000000000000000000000000000000/idp/wstrust.
- **STSPolicyID**
  Should be left blank when connecting to STS.
- **identityProviderType**
  Defaults to ADFS. This setting should be left unchanged when connecting to STS.

Using SSL

Signed certificates: EAM opens http connections to STS and to other modules within EAM. If these connections employ SSL, the certificate returned by the server (STS, or the instance of Apache installed with EAM) must be signed by a CA that can be validated using the certificate chains present in the EAM cacerts file (located in <eamhome>/java/jre/lib/security)..

- The certificate returned by STS will be validated against cacerts if any of the following EAM functionality is used:
    - Web services, including all connector requests (mobile, databridge, etc.)
    - Electronic signatures
    - Advanced reports
    - Ensure that LDAP Authentication Store is configured in STS for all of the above services to work.
      See the *Infor Operating Service Installation Guide* for configuring LDAP authentication store in STS.
- The certificate returned by STS is part of the WS-FED metadata xml obtained from Download Federated Metadata section in the STS Admin UI. https://<sts-server>:<port>/inforsts/rest/metadata/00000000000000000000000000000000/wsfed/idp

Configuring EAM With Infor OS Manager

EAM application must be configured with Infor OS Manager and not from STS Admin UI so that all the required SAML Claims are issued for the user. This configuration must be done with sp.xml file

generated under <ssoclienthome> in the EAM application. For details regarding the configuration of an application with Infor OS Manager, see the *Infor Operating Service Installation Guide.*

Creating a custom logon module

This section describes in detail how to develop a custom logon module to connect HxGN EAM to a custom external Single Sign-On (SSO) server. When HxGN EAM is configured to use an SSO server, all authentication requests will be routed to the SSO server; the HxGN EAM logon screen will not be displayed.

Single Sign-On processing flow

The flow for single sign-on is as follows:

1. A user attempts to log onto HxGN EAM from a browser.
2. HxGN EAM detects that external logons are enabled (the LGNEAM install parameter is set to 'EXTERN').
3. HxGN EAM retrieves the URL of the SSO server from the logonURL property in the sso configuration file and redirects the browser to this URL.The URL that should be used in returning the user to HxGN EAM is passed to the SSO server on the query string.
4. The SSO server authenticates the user by some means, typically by prompting the user for a logon id and password.
5. If the user is authenticated successfully, the SSO server will redirect the browser back to HxGN EAM. The SSO server may include any parameters it deems necessary on the query string.
6. HxGN EAM creates an instance of the authentication handler class specified in the authenticationHandler property of the sso configuration file. All query string parameters passed from the SSO server to HxGN EAM will be passed to the handler
7. The authentication handler decides whether the user should be allowed to log in based on the query string parameters. The authentication handler may choose to contact the SSO server to validate the parameters passed on the query string.
8. If the authentication handler determines that the logon should proceed, it returns the HxGN EAM userid to the system.
9. The usual HxGN EAM logon process is followed, except that the user's password will not be checked against the HxGN EAM database.

If all logon requirements are satisfied but a record for the user does not exist in the HxGN EAM database, the following occurs:

- If a securityHandler class is defined in the sso configuration file, the handler class will be invoked.

  The purpose of the securityHandler is to map a userID/tenant/product to an HxGN EAM role name.
- If a securityHandler class is defined and it returns a role that exists in the HxGN EAM database, a new user record will be created based on that role. In all other cases, the logon attempt will fail.

Requirements for a custom SSO implementation

Modifications to HxGN EAM include the following:

- An authentication handler must be written in Java and added to the HxGN EAM deployment.
- A security handler may optionally be added to the deployment as well.

An SSO server can be integrated with HxGN EAM if it is able to perform the following actions:

- Handle a browser-based authentication request.

  Since HxGN EAM does not collect authentication credentials from the user, the SSO server must be able to collect from the user whatever information it needs for authentication.
- Redirect a browser back to HxGN EAM based on a dynamically specified URL (see below).
- Communicate with the authentication and security handlers.

  These handlers will presumably connect to the SSO server by some means to obtain the information they require. The SSO server must provide the necessary interface.

The AuthenticationHandler interface

To connect to a custom SSO server, a Java class that implements the AuthenticationHandler interface must be added to the HxGN EAM runtime environment.

The fully qualified name of the AuthenticationHandler interface is com.dstm.mp.businessprocess.base.logon.handlers.AuthenticationHandler.

The interface may be implemented directly or by subclassing a Java class that implements the interface. A class is provided that implements the interface and provides defaults for some of its methods (AuthenticationHandlerImpl).

The AuthenticationHandler interface contains five methods. The most important of these are the two authenticate methods discussed below.

String authenticate (String ticket, Map properties, String tenantID, String productID)

The string authenticate is called to validate the logon attempt after HxGN EAM receives a redirect from an SSO server.

- **Ticket parameter:**
  - The SSO server may optionally pass a parameter on the query string called ‘ticket’. The ticket contains a token that the authentication handler can pass back to the SSO server to verify the validity of the logon request.
  - If the SSO server does not pass a ticket parameter, ticket will be set to ‘INVALIDTICKET’. The logon may nevertheless be successful: the authentication handler may choose to ignore the ticket parameter and validate the logon request based on some other value in the query string.
- **Properties parameter**: a map containing all the query string parameters passed from the SSO server to HxGN EAM. The map may contain additional values supplied by HxGN EAM.
- **UserID parameter**: the HxGN EAM user ID
- **Password parameter:** the password to be used for authentication
- **tenantID parameter:** the current tenant
- **productID parameter:** one of the following:
  - connector
  - databridge
  - mobile
  - eam
- Return value
  - If the logon attempt is accepted, return the HxGN EAM userID
  - If the logon attempt is rejected, throw one of the checked exceptions

Void authenticate (String userID, String password, String tenantID, String productID)

Called to validate the logon attempt when HxGN EAM already knows the username/password.

Specifically, this method is called for:

- All Connector logons
- Logons to the Advanced Reporting Server
- Electronic signature validation

- **UserID parameter**: the HxGN EAM user ID
- **Password parameter:** the password to be used for authentication
- **tenantID parameter**: the current tenant
- **productID parameter:** one of the following:

    - connector
    - databridge
    - mobile
    - eam
- **Return value:**
    - If the logon attempt is rejected, throw one of the checked exceptions.

Security handler

Requirements for the Security Handler

A Security Handler must implement the SecurityConfigHandler interface.

- The fully qualified name of the interface is com.dstm.mp.businessprocess.base.logon.handlers.SecurityConfigHandler

The SecurityConfigHandler interface contains one method: String resolveRole.

String resolveRole (String userID, String tenantID, String productID)

Called when the user has been authenticated by the SSO server, but no user record exists in the HxGN EAM database. The return value should be the name of a valid HxGN EAM role.

Compiling Authentication and Security Handlers

Handlers should be compiled using the same version of Java shipped with HxGN EAM.

In order to compile a handler, at a minimum the following jars must be on the classpath:

- mp_subsystem.jar
- mp_common.jar

These jar files can be found in the following location: <eamhome>\depconfig\skeletons\jboss\lib

Deploying Handlers

The handlers and all supporting classes should be packaged in a new jar file. If the handlers utilize xml configuration files or properties files, these should also be included in the jar file.

These are the steps to deploy the custom jar file:

JBoss

1 Copy the jar file to this location: <eamhome>\overrides\customerjars\

2 Redeploy

Adding jar files to HxGN EAM

If a handler makes use of third-party jar files (for xml parsing, http transmission, etc.) it should use the jars that ship with HxGN EAM. Third party jar files should not be added to HxGN EAM; such jar files may conflict with the installed libraries and break existing functionality.

HxGN EAM functionality and handlers

There are several classes/libraries available for use by a handler:

- **Logging:** a handler may wish to write to the standard application log files (see the sample and the Javadocs for more information).
- **External logon configuration:** a class called LogonConfig provides access to the settings in the sso configuration file; see the Javadocs for more information.
- **Third party libraries**: several libraries are available for use in communicating with the SSO server. In particular, some of the Apache Axis libraries are available; see the following documentation on the sample application for more information.

Authentication handler sample

A sample authentication handler is included; it can be found in the ‘sourcecode’ directory at com.infor.ssocustom.handlers.WSAuthenticator.

This handler illustrates the following:

- How to implement all methods in the AuthenticationHandler interface
- How to make use of the HxGN EAM logging mechanism
- How to access a properties file included in a custom jar file
- How to connect to an SSO server using web services

Logging integration

HxGN EAM uses the standard log4j logging mechanism. The DSTMLogger class used in the sample is a thin wrapper around log4j.

See the Javadocs for the methods available in the DSTMLogger class. The log4j configuration file can be modified to log information from the custom classes.

Web service connections to the SSO Server

WSAuthenticator uses web services to connect to the SSO Server. It is not necessary for handlers to connect to the SSO server via web services—WSAuthenticator simply illustrates one way the conversation between handler and server can be managed.

These are some of the highlights of the Web Service implementation in WSAuthenticator:

- The current version of HxGN EAM uses Apache Axis version 2.1.6.4. If Axis is used to generate web service artifacts, it is recommended that this version be used.
- The Axis Java2WSDL utility was used to generate a wsdl based on the Java class that represents the interface to the SSO server.
- The Axis wsdlgen utility was used to generate Java client classes from the wsdl. These classes are in the package com.infor.sso.surrogate.
- WSAuthenticator communicates with the SSO server by means of these generated classes.