How to Integrate
Workload Automation (WA)
with
Security Assertion Markup Language
(SAML)

IBM/HCL Workload Automation

Sajjad M. Kabir
Solutions Architect
IBM/HCL Lab Services

Version: 1.0
Date: Feb 03, 2025

1 Document History

1.1 Document Location

This is a snapshot of an on-line document. Paper copies are valid only on the day they are printed. Refer to the author if you are in any doubt about the currency of this document.

1.2 Revision History

The following is a revision history of the document.

Revision Number	Revision Date	Summary of Changes	Changes Marked
1.0	Feb 03, 2025	Initial version	

1.3 Approvals

This document requires following approvals.

Name	Title

Contents

2 Introduction

With the advent of cloud computing offerings by various vendors, such as, Amazon Web Services, Microsoft Azure, and Google Cloud Computing, came the need to integrate with authentications services on the cloud as well. Security Assertion Markup Language (SAML) and OpendID Connect (OIDC) and prime examples of these authentication services, which provide a federated authentication mechanism without migrating the user repositories to the cloud. This makes it attractive for corporations to take advantage of the service without taking on any additional complexities or security risks.

This document provides a procedure to integrate IBM/HCL Workload Automation with SAML.

3 Overview of SAML

SAML stands for Security Assertion Markup Language. It's a technical standard that allows different systems to securely exchange information about the identity of the user.

SAML can be considered as a middleman. Instead of each application (Service Provider, SP) having to build its own authentication mechanism validating a user with the username and password, it trusts a central Identity Provider (IDP) that assures the identity of the token. This assurance is provided through an Assertion Token, which is a digital document that says "Yes, this person is John Doe, and they're authorized to access this application."

SAML also enables Single Sign-On (SSO). A user only needs to login once and the SAML token issued by the IP can be used in other applications, SPs, without having to login again with the user’s credentials.

Here's how it works:

1. A user accesses an application
2. The application redirects the user to the Identity Provider’s login page
3. The user logs in to the Identity Provider
4. The Identity Provider creates a SAML assertion token and sends it to the application
5. The application trusts the assertion and grants the user access

4 Integrating Workload Automation with SAML

The following is a high-level process:

1. Register the DWC as an SP (Service Provider) to the IDP (Identity Provider)
2. Provide the metadata of the SP to the IDP
3. Receive the metadata of the IDP
4. Specify the claim names provided by the IDP in the SP configuration
5. Attempt to login to the DWC

4.1 Integration Procedure

Follow the steps below to integrate the DWC with SAML.

1. Collect the following information from the Identity Provider

   a. nameIDFormat

   The nameIDFormat is usually set to "unspecified". If it is not, the following error can be seen in messages.log:

   urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified.

   b. userIdentifier

   The userIdentifier is usually set to "email". If it is not, the following error can be seen in messages.log:

   <samlp:StatusMessage>Cannot provide requested name identifier with format urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress for the given subject</samlp:StatusMessage>

   Ask the IDP to set the NameIDPolicy format to urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress and change the NameID value from UUID to email address based on the DWC requiring NameID format as email address.

   c. groupIdentifier

   The groupIdentifier must be is set to a value that matches the value in IDP. If the user belongs to multiple groups, the IDP must provide the groups in the following format:

   "groups": ["group1","group2"]

2. Create a config file similar to other authentication mechanism config files. The following is a sample. The client id used in this sample is called **waSP**, which will be used in the URL in a later step. The values for the above attributes may not be available yet, which can be updated later. This is required to generate the SP metadata file. Place this file in the overrides dir:

   cd <DWC_Install_Dir>/usr/servers/dwcServer/configDropins/overrides
   vi saml_authentication_config.xml

```
<server>
      <featureManager>
            <feature>samlWeb-2.0</feature>
      </featureManager>
```

```
    <authFilter id="urlFilter">
        <requestUrl id="Intercept" urlPattern="/console" matchType="contains"/>
        <requestUrl id="doNotIntercept" urlPattern="/login.jsp" matchType="notContain"/>
    </authFilter>

    <samlWebSso20
        id="waSP"
        idpMetadata="/opt/dwc/usr/servers/dwcServer/resources/security/idpMetadata.xml"
        keyStoreRef="twaKeyStore"
        spAlias="server"
        sslRef="defaultSSLSettings"
        authType="saml"
        authFilterRef="urlFilter"
        realmName="WASAML"
        nameIDFormat="unspecified"
        userIdentifier="email"
        groupIdentifier="group"
        mapToUserRegistry="No">
    </samlWebSso20>
</server>
```

3. Restart the DWC

   cd <DWC_Install_Dir>/appservertools
   ./stopAppServer.sh
   ./startAppServer.sh

4. Verify that the SP metadata is available at the following link. If it is, provide the link to the IDP so that they can retrieve the SP metadata and register it as a client

   https://<dwc_server_fqdn>:<port>/ibm/saml20/**waSP**/samlmetadata

5. Once the registration is complete (also known as federation), the IDP provides a link to download the IDP metadata. Go to the URL and save the contents in a file named exactly as shown below in the following dir. This was already provided in the config file in an earlier step.

   <DWC_Install_Dir>/usr/servers/dwcServer/resources/security/**idpMetadata.xml**

6. For Kubernetes/Helm charts deployments, the IDP Metadata file can be placed in the overrides dir by creating a configMap with a name that doesn't end in (so that WLP doesn't try to process it). In such a case, the file, saml_authentication_config.xml, above should have the path in the attribute, idpMetadata, pointing to the overrides dir.

7. If the values assigned to the attributes earlier are incorrect, update the config file with the correct values and restart the DWC WLP process.

8. If the DWC and MDM are deployed on the same server, no changes to the default configuration file, **jwtsso.xml**, in both DWC and MDM's overrides dir is required.

9. If the DWC and MDM are deployed on different servers, the attribute, **jwksUri**, must be updated in the default configuration file in the MDM's overrides dir, jwtsso.xml, to point to the respective URL of the DWC.

10. Determine what the scope and claim of the new JWT be and assign the corresponding values to those two attributes. MDM only needs two claims, user ID and group, from the new JWT. Whichever claims represent those two properties, specify them in the config file below.

11. If the following file doesn't exist, jwtsso.xml file, create it as follows:

cd <**DWC**_Install_Dir>/usr/servers/**dwcServer**/configDropins/overrides
vi **jwtsso_dwc.xml**

```
<server>
     <featureManager>
<feature>jwtSso-1.0</feature>

</featureManager>

      <jwtBuilder id="jwtBuilder"
            jwkEnabled="false"
            keyStoreRef="twaKeyStore"
            keyAlias="server"
            issuer="console"
            audiences="iwausers"
            scope="openid profile email roles"
            claims="sub,roles">
      </jwtBuilder>

       <mpJwt id="mpJwt"
            jwksUri="https://localhost:9443/jwt/ibm/api/jwtBuilder/jwk"
            issuer="console"
            audiences="iwausers"
            sslRef="twaSSLSettings"
            clockSkew="5m"/>

      <httpSession securityIntegrationEnabled="false" invalidationTimeout="12h" />

      <jwtSso cookieName="iwajwt" jwtBuilderRef="jwtBuilder"/>


</server>
```

12. If the following file doesn't exist, jwtsso.xml file, create it as follows:

    cd <**MDM**_Install_Dir>/usr/servers/**engineServer**/configDropins/overrides
    vi **jwtsso_mdm.xml**

```
<server>

      <authFilter id="doNotIntercept">
            <requestHeader id ="bearer" name="Authorization" value="Bearer"
matchType="contains"/>
      </authFilter>

      <mpJwt id="mpJwt"
            jwksUri=https://pristella.raleigh.ibm.com:9443/jwt/ibm/api/jwtBuilder/jwk
            issuer="console"
            audiences="iwausers"
            sslRef="twaSSLSettings"
            clockSkew="5m"
            authFilterRef="doNotIntercept"/>
</server>
```

13. Notice that the DWC is on a separate server, pristella.raleigh.ibm.com, indicated by **jwksUri**.
14. If the IDP uses a custom certificate, retrieve the public certificate and import into the DWC TWSServerTrustStore.
15. Restart the DWC

cd <DWC_Install_Dir>/appservertools
./stopAppServer.sh
./startAppServer.sh

16. Attempt to login
17. If login fails, refer to DWC WLP messages.log for troubleshooting

4.2 Understanding SAML Assertion Token

The following is an explanation of how to understand the syntax of the SAML Token.

1. The usual header of an assertion token is as follows:

   urn:oasis:names:tc:SAML:1.1

   URN (Uniform Resource Name) namespace that is engineered by OASIS. It is a naming standard.

   OASIS Organization for the Advancement of Structured Information Standards for naming persistent resources published by OASIS (such as OASIS Standards, XML (Extensible Markup Language) Document Type Definitions, XML Schemas, Namespaces, Stylesheets, and other documents).

   TC Technical Community

2. He following is an example of a common name identifier format (**nameid-forma**t) of a currently authenticated user, which is set to the **email address** of the user:

   urn:oasis:names:tc:SAML:1.1:**nameid-format**:**emailAddress**

3. The following is a full SAML Assertion for an authenticated user

```
<saml:Assertion>
   <saml:Subject>
      <saml:NameID Format="urn:oasis:names:tc:SAML:1.1:nameid-
format:emailAddress">sajjad.kabir@kics.com</saml:NameID>
      <saml:SubjectConfirmation Method="urn:oasis:names:tc:SAML:2.0:cm:bearer">
         <saml:SubjectConfirmationData InResponseTo="_qoXeXvVwvrA0vl6uRIsgnXCJHrZjSwlB"
NotOnOrAfter="2024-10-18T17:51:09.102Z"
Recipient="https://waconsole.schedaks.az.fiscal.treasury.gov/ibm/saml20/iwaSP/acs"/>
         </saml:SubjectConfirmation>
   </saml:Subject>

   <saml:Conditions NotBefore="2024-10-18T17:41:09.102Z" NotOnOrAfter="2024-10-18T17:51:09.102Z">
      <saml:AudienceRestriction>
      <saml:Audience>https://waconsole.schedaks.az.fiscal.treasury.gov/ibm/saml20/iwaSP</saml:Audience>
      </saml:AudienceRestriction>
   </saml:Conditions>

   <saml:AuthnStatement AuthnInstant="2024-10-18T17:46:09.100Z" SessionIndex="J4-xgntE9hAZijJ-
mic8Wt7.1bc">
      <saml:AuthnContext>
            <saml:AuthnContextClassRef>http://idmanagement.gov/ns/assurance/ial/3/aal/3
            </saml:AuthnContextClassRef>
         </saml:AuthnContext>
      </saml:AuthnStatement>

   <saml:AttributeStatement>
      <saml:Attribute Name="role" NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified">
         <saml:AttributeValue xsi:type="xs:string" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">Administrators</saml:AttributeValue>
```

```
      </saml:Attribute>

      <saml:Attribute Name="given_name" NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-
format:unspecified">
         <saml:AttributeValue xsi:type="xs:string" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">Sajjad</saml:AttributeValue>
      </saml:Attribute>

      <saml:Attribute Name="middle_name" NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-
format:unspecified">
         <saml:AttributeValue xsi:type="xs:string" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">M.</saml:AttributeValue>
      </saml:Attribute>

      <saml:Attribute Name="family_name" NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-
format:unspecified">
         <saml:AttributeValue xsi:type="xs:string" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">Kabir</saml:AttributeValue>
      </saml:Attribute>

      <saml:Attribute Name="uuid" NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified">
         <saml:AttributeValue xsi:type="xs:string" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">66982463-20e5-48b7-aa18-
fc30f8435dba</saml:AttributeValue>
      </saml:Attribute>

      <saml:Attribute Name="email" NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified">
         <saml:AttributeValue xsi:type="xs:string" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">Sajjad.Kabir@kics.com</saml:AttributeValue>
      </saml:Attribute>
   </saml:AttributeStatement>
</saml:Assertion>
```

5 Author's Bio

Sajjad Kabir is a Solutions Architect and a certified Workload Automation consultant with over 30 years of experience. He has a B.Sc. in Computer Systems Engineering from Western Michigan University, Kalamazoo, MI. Sajjad started working with IBM Workload Automation in 1998 while on assignment in IBM Singapore. He has been assisting clients deploy Workload Automation solutions worldwide. He has worked with clients of all sizes and complexities. He has published an IBM Redbook and numerous articles and blogs. He loves to work with people, especially who have just started their endeavors with workload automation. With the transition from IBM to HCL, he has mentored HCL consultants and continued to assist clients with implementing automation solutions on-prem and in the cloud. Sajjad loves to travel, enjoys ethnic cuisines, and SCUBA diving in exotic places around the world.