Recip-e Connector Session Management:

Migration Guide from 1.6.x to 1.7.x or above

1. Purpose of this document

This document describes the session management of the several versions of the Connector and can be used to migrate from Recip-e Connector 1.6.x to 1.7.x or above.

The session management has been reworked in version 1.7.x and has not been changed afterwards.

2. Session management in 1.6.x

In Recip-e Connector 1.6.x, the CommonIntegrationModule contains the operations createSession/ createFallbackSession, loadSession/unloadSession and assertValidSession. When using the 1.6.x Connector, the External Software Integrator is responsible to handle the session manually by calling there these operations.

Below an example to create a session:

```
//Create a session with EID

String samlToken = commonIntegrationModule.createSession();

//Or create a fallback session (by using the eHealth certificates)

String  samlToken = commonIntegrationModule.createFallbackSession(…)

//Load the session into the commonIntegrationModule

commonIntegrationModule.loadSession(samlToken);
```

Once the session is created, the External Software Integrator is able to call the Recip-e operations. The example below shows the createPrescription-operation:

```
//Validate the session

commonIntegrationModule.assertValidSession();

//Call the createPrescription-operation

CreatePrescriptionResponse response =
RecipePrescriberServiceImpl.getInstance().createPrescription(…);
```

If the SAML-token is elsewhere required, f.e. to secure the request, the External Software Integrator can verify that there is a valid session by:

```
//Validate the current session

Session.getInstance().hasValidSession();
```

The SAML-token is available by calling:

```
//Obtain the SAML-token

Session.getInstance().getSession().getSAMLToken();
```

3. Session management as of 1.7.x

As of version 1.7.x, a SessionUtil/SessionCache/SessionValidator are available to create/maintain/validate a session.

Before calling a Recip-e operation, first fetch the current session:

```
//Fetch the current session

SessionItem cachedSession = SessionCache.getInstance().get(configFile, configValidationFile);
```

Next, validate it is still a valid session:

```
//Validate the session

SessionValidator.isValidSession(cachedSession)
```

If the session is still valid, load it into the SessionUtil:

```
//Store the session in the SessionUtil

SessionUtil.loadExistingSession(configuration, cachedSession.getSAMLToken());
```

If the session is no longer valid, create a new session, f.e.:

```
//Request a new session
Properties configuration = new PropertyHandler(configFile, configValidationFile)
SessionUtil.createSession(SessionType.FALLBACK_SESSION, configuration, null, null);
```

And store it in the SessionCache:

```
//Store the session in the SessionCache
SessionCache.getInstance().put(configFile, configValidationFile, sessionItem);
```

Now, the createPrescription-operation can be called:

```
//Create a prescription
getPrescriberIntegrationModule().createPrescription(true, patientId, getTestPrescriptionPatientid(), "P1");
```

In order to create a SAML-secure request at a later phase when building the request, the following operations can be used:

```
//Get the SessionItem
final SessionItem sessionItem = Session.getInstance().getSession();
//Validate the SessionItem
SessionValidator.assertValidSession(sessionItem);
//Get the SAMLAssertion from the SessionCache
sessionItem.getSAMLToken().getAssertion()
//Get the HOK from the SessionCache:
sessionItem.getHolderOfKeyCredential()
```

4. Document revision

Version Number	**Date**	**Author**	**Changes (content)**
Version 1.1	2020/01/21	Recip-e Development Team	Initial version