Configure your ABAP Development System for Development of HDI Objects

Version 1.3
### Document History

<table>
<thead>
<tr>
<th>Document Version</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.3</td>
<td>Enhanced chapter <strong>Introduction and Preparation</strong> and <strong>General Concepts</strong></td>
</tr>
</tbody>
</table>
| 1.2              | Chapter **Communication from ABAP to XSA failed** extended with additional troubleshooting  
Chapter **Configuring Secure Connection from ABAP to XSA** extended  
Chapter **Error in existence check for XSA space** added;  
Chapter **Action GET_UPS_ID (GET)** was canceled added; minor corrections |
| 1.1              | Chapters **Enable transaction OA2C_CONFIG and SAP Web IDE for SAP HANA keeps loading** added; minor corrections |
| 1.0              | First official release of this document |
TABLE OF CONTENTS

1 INTRODUCTION AND PREPARATION .................................................................................. 5
1.1 General Concepts.......................................................................................................... 5
1.1.1 Development Process ............................................................................................... 7
1.1.2 Configuration in a Nutshell ...................................................................................... 8
1.2 Prerequisites.................................................................................................................. 9
1.3 Users ............................................................................................................................ 9
1.4 Further Information ...................................................................................................... 10
2 CONFIGURATION ............................................................................................................. 10
2.1 Checking if HDI is Enabled............................................................................................ 11
2.2 Preparing the Communication from XSA to ABAP ....................................................... 12
2.2.1 Creating Database Procedures Required by AS ABAP ............................................. 12
2.2.2 Creating an SAP HANA Role .................................................................................. 15
2.3 Preparing the Communication from ABAP to XSA ....................................................... 17
2.3.1 Creating Role Collections in XSA ........................................................................... 17
2.3.2 Create User HTA_ADMIN_<ABAP_SID> in XSA .................................................... 23
2.3.3 Creating Destinations in ABAP .............................................................................. 27
2.3.4 Configuring the Communication from ABAP to XSA .............................................. 34
2.4 Configuring the Space .................................................................................................. 39
2.4.1 Creating XSA Space ABAP_SAPAT5 ................................................................. 39
2.4.2 Enable Development in Space ABAP_SAP<ABAP_SID> ......................................... 43
2.5 Maintaining the Web IDE URL on ABAP .................................................................... 44
2.6 Creating the Shared Workspace .................................................................................. 47
2.7 Enable an ABAP Developer for HDI Object Development ......................................... 49
2.7.1 Assigning Authorizations to the Developer on ABAP ............................................. 49
2.7.2 Creating a Developer User on XSA ....................................................................... 49
2.7.3 Assigning Developer Role in XSA ......................................................................... 51
2.7.4 Adding Developer User to Space .......................................................................... 52
2.7.5 Assigning the XSA User to the Web IDE Shared Workspace ................................. 55
3 CHECKING THE CONFIGURATION (OPTIONAL).......................................................... 56
4 TROUBLESHOOTING ........................................................................................................ 56
4.1 Cannot Select ANONYM SSL Client (Anonymous) from the Drop-Down for SSL Certificate ................................................................. 57
4.2 Error when Executing Connection Test for Destination .............................................. 57
4.2.1 HTTPIO_PLG_CANCELED .................................................................................... 57
4.2.2 SSL Handshake Fails............................................................................................. 59
4.2.3 Connection to <host>;<port> failed: NIEHOST_UNKNOWN(-2) .............................. 59
4.2.4 Connect to <host>;<port> failed: NIECONN_REFUSED(-10) ............................... 59
4.2.5 No Logon-pop-up ............................................................................................... 60
4.2.6 Communication from ABAP to XSA failed .......................................................... 60
4.3 Enable transaction OA2C_CONFIG ............................................................................. 61
4.4 Error when Requesting OAuth 2.0 Tokens .................................................................. 62
4.4.1 HTTP Failure, Processing Failed, Invalid State, Invalid Timeout or Others........... 62
4.4.2 Client Authentication Failed (e.g., Unknown Client, no Client Authentication Included, ...) ................................................................. 63
4.4.3 The Sever is Refusing to Respond to the Request (HTTP 403 – Forbidden) .......... 63
4.4.4 Requested Resource could not be found (HTTP 404 – not found) ......................... 63
4.5 Error in existence check for XSA space ....................................................................... 63
4.5.1 Error when executing GET_SPACE_ID command (http code is 0) ..........................................63
4.6 Space Enablement UI Does not Open, Error Message: Forbidden ........................................64
4.7 Error when Creating Shared Workspace .............................................................................64
4.7.1 Action CREATE HTTP REST CLIENT was canceled (http status code 0) .......................64
4.7.2 Action CREATE WORKSPACE(POST) was Canceled (Http Status Code 404) .............65
4.7.3 Action CREATE WORKSPACE(POST) was Canceled (Http Status Code 500) ...............65
4.7.4 Invalid action ..................................................................................................................65
4.7.5 OAuth 2.0 Client Profile SAP_HTA_HDI is not Assigned to an OAuth 2.0 Client ...............66
5  APPENDIX ..............................................................................................................................66
5.1 Script for Creating Database Procedures Required by AS ABAP .......................................66
5.2 Script for Creating an SAP HANA Role .............................................................................70
5.3 Configuring Secure Connection from ABAP to XSA ..........................................................74
5.4 Using the Report SCTS_HTA_CONFIGURATION .................................................................85
5.5 Abbreviations .......................................................................................................................85
This guide provides you with an overview on SAP HANA Transport for ABAP for SAP HANA Deployment Infrastructure (HTA for HDI). In addition, it explains the prerequisites that must be fulfilled on your development systems before you can use HTA for HDI. Additional configuration is required. This guide explains in detail what has to be done.

1 INTRODUCTION AND PREPARATION

This guide is only relevant for development systems (so systems where you would like to create objects and export them). None of the configuration steps described below has to be executed on test or production systems.

To be able to develop and transport HDI objects in the context of ABAP, you only need a SAP Web IDE for SAP HANA (Web IDE) connected to the ABAP development system. For this, you only need one XSA server that has the Web IDE installed. ABAP creates the HDI containers on the target systems that are part of the transports requests. Therefore, XSA is not needed on any target system (as long as you do not develop native XSA applications – to be able to transport these native XSA applications, you would have to configure CTS+ as HTA for HDI only supports transport of HDI objects).

Please check SAP Note 2569651 - Configure your ABAP Development System for Development of HDI Objects before you start the configuration. The note contains updates and information that you need during the configuration.

1.1 General Concepts

HDI (SAP HANA Deployment Infrastructure) is a service layer of the SAP HANA database that simplifies the deployment of SAP HANA Database artifacts.

HDI artifacts are developed using Web IDE that is available on premise as an SAP HANA XS, Advanced Model (XS advanced / XSA) application. Each ABAP development system that is used to develop or change HDI objects needs an instance of Web IDE assigned to it. Multiple ABAP systems can share an instance of the Web IDE. At runtime, no Web IDE instance is required. Therefore, test or production systems do not need a Web IDE and therefore no configuration.

To be able to develop HDI objects, you have to use the Web IDE (in the past – if you worked with SAP HANA Repository – the SAP HANA Studio or SAP HANA Web-based Development Workbench was used for this purpose).

As you might know ABAP developers work on one system (not locally). In Web IDE, this concept is realized using a shared workspace where all developers connect to and where all developed objects are stored.

1 NOTE

Using an ABAP system running on SAP HANA is absolute basic prerequisite for using HDI. If your ABAP system does not run on SAP HANA, you cannot use any of the functionality (HDI or HTA for HDI) described in this guide.

The following picture shows the basic elements of ABAP and SAP HANA that are involved in HTA (SAP HANA Transport for ABAP) for HDI including the system IDs that we will use as example in this guide. The figure shows the systems (ABAP and XSA) that are involved in the development activities. Target systems (like test or production) only consist of an AS ABAP (like AT5) and its SAP HANA server (like HDB) used as database for ABAP. The next two chapters will explain the role and usage of these elements in the development process and how they have to be configured.
Destination required to create a shared workspace, add users to this workspace, create projects, and copy HDI objects from the shared workspace to HTA Repository in ABAP and vice versa. Later in the configuration, you will create this destination and name it XSA DEVX.
Destination needed by ABAP to automatically create external services (user-provided services) – see also connection 3. Later in the configuration, you will create this destination and name it XSA_ADMI.

These connections are used by:
- DI Core (DevX) when a build in Web IDE is triggered to deploy the HDI artifacts to the HDI container in the database of the ABAP system.
- Modeling Tools in Development Perspective in Web IDE (e.g. Calculation View Editor or Synonym Editor) to retrieve and display the definition of existing catalog objects (e.g. views, tables, …) of a schema/HDI container in the database of the ABAP system.
- The Database Explorer of SAP Web IDE for SAP HANA so that Developers can test their HDI objects (e.g. Calculation Views)

These connections are created automatically by HTA as an external service (user-provided service) in the XSA space ABAP_SAP<SID> when a developer creates the first HDI namespace for a container in ADT or when a developer does the first check out of existing HDI objects of a container in ADT.

### 1.1.1 Development Process

In this guide, you will configure your system so that a developer can afterwards develop native SAP HANA artifacts (via HDI design-time objects) and use them in the context of ABAP. This is needed if the development options offered by AMDP (ABAP Managed Database Procedures - framework for managing and calling stored procedures or database procedures in AS ABAP) or ABAP CBS (ABAP Core Data-Services make it possible to define semantic data models on the central database of the AS ABAP) are not sufficient and you need to develop other objects like e.g. calculation views in SAP HANA that you need to use in ABAP.

Once the system is configured for HDI-object development, the development process for HDI-objects looks like this:

1. The developer (or an administrator) creates an HDI container in transaction SCTS_AMHC on the system AT5 in our example (there should not be that many containers in one system, so this should be done somehow centrally and not by each developer). A container is a triple of schema/user pairs: for runtime objects, design-time object storage, and object owner.
2. The developer creates a namespace in an HDI container using ABAP development tools (ADT / also known as ABAP in Eclipse – AiE). This will automatically create a project in the shared workspace on XSA (system ZDM on the left side in our example), this communication is illustrated by 1 in the picture above (the destination is named XSA_DEVX). All users develop in the shared workspace. In addition, an external service that allows the communication from XSA to the HDI container in the SAP HANA database of AS ABAP is automatically created. Destination XSA_ADMI will be used for this 2 in the picture above.
3. The developer develops HDI objects in SAP Web IDE (a browser application on his client) and deploys them via the deployment infrastructure (DI Core in XSA) into the HDI container in the SAP HANA database on the ABAP server. This is done via the external service that was created in the previous step.
4. The developer checks in his objects into ABAP using ADT. During the check in the developer is also asked to collect the objects in a transport request in CTS. The objects are now copied from the shared workspace on XSA to the HTA repository on ABAP. Destination XSA_DEVX is used for reading the files from the shared workspace. As a result, the HTA repository contains HDI objects and ABAP managed HDI containers.
5. The developer releases the transport request.
6. (not part of the picture above) The administrator imports the transport request into the target system. Now, the objects that are part of the transport request are imported into the HTA repository during the import step Import and deployed from the HTA repository to the HDI container during the import step SAP HANA Deployment (in older releases, the step is named SAP HANA Repository Deployment). For the import process, there is no SAP Web IDE for SAP HANA required on the target system.
7. Before you can start developing, you have to set up all the required communications described above. This guide explains in detail how this is done.
1.1.2 Configuration in a Nutshell

In this guide, we will use the landscape shown in the picture in chapter 1.1 General Concepts as an example. We will use one system named ZDM. On this SAP HANA system, XSA and the SAP Web IDE for SAP HANA are installed.

In addition, we need AS ABAP which runs on an SAP HANA database. In our example, this system is named AT5.

HDI runs on SAP HANA of AT5 – not on ZDM

We provide a program named SCTS_HTA_CHECK_CONFIGURATION. This program guides you through the configuration. It includes test for most of the configuration steps described in this guide. We recommend that you use this program to check which configuration steps have to be done and whether they were successful. All configuration chapters in this guide describe how to use the report for the respective step and how to deal with possible errors.

First step is to check that HDI in general is enabled in your system. In most cases, this should have been done by SUM/SWPM – see chapter Checking if HDI is Enabled.

As a next step, you have to execute two scripts that will create procedures to be able to create technical users and a role in SAP HANA database of AS ABAP. These are required later on during development to automatically create external services in XSA. These external services are later used by DI Core to communicate with the HDI container in the SAP HANA database of AS ABAP. Each external service contains a technical user that needs a dedicated role in HANA database of AS ABAP.

After that, you need to make sure that the communication between ABAP (AT5 in our example) and XSA / SAP Web IDE (ZDM in our example) is possible. For this, two http(s)-destinations need to be established:

- XSA_DEVX: ABAP needs to be able to communicate with DI Core (DevX) to create a shared workspace, add users to this workspace, create projects, retrieve objects from the shared workspace to ABAP (Check-in) and push objects from ABAP to the shared workspace (Check-out).
- XSA_ADMI: ABAP needs to be able to communicate with the XSA controller to create external services. This service is created automatically by HTA when you check out objects out of an existing HDI namespace or when you create a new HDI namespace. The service contains the connection data for the HDI container of the SAP HANA database of AS ABAP including a technical user.

The destinations XSA_ADMI and XSA_DEVX are configured in chapter Creating Destinations in ABAP. As XSA only allows connections with OAuth2 authorization the communication from ABAP to XSA requires the configuration of an OAuth2 client on AS ABAP This is done in chapter Configuring the Communication from ABAP to XSA.

The communication via these destinations requires a user. The respective technical user is created in chapter Create User HTA_ADMIN_<ABAP_SID> in XSA.

The shared workspace is created in chapter Creating the Shared Workspace

The program SCTS_HTA_CHECK_CONFIGURATION is designed to guide you through the configuration steps. It shows if a configuration has already been completed and what has to be done next. In case of errors, the report gives hints on how to solve them. The different test in the program are named like the chapters in this guide.

All configuration actions described in this guide need to be performed on development systems only. That is, for systems where you want to develop and change HDI objects with SAP Web IDE.

Transports are later on executed via the known mechanisms of TMS / CTS. You do not need to configure any additional transport routes or systems in transaction STMS.

\* NOTE

In this guide, we assume that the transport routes between the different ABAP systems are already in place. The required configurations are not described in this guide. If you need information on how to do this, refer to the SAP Help Portal at https://help.sap.com/viewer/CTS75SP06/44b4a09a7acc11d1d1899e0000e829fbdb.html 'Configuring TMS'
1.2 Prerequisites

- In this guide, we assume that you have configured transport routes for your ABAP systems in the Change and Transport System (transaction STMS). There are no additional configurations required concerning transport routes if you like to use HTA for HDI. Details about the change and transport system and how to set up transport routes are available on the SAP Help Portal at https://help.sap.com/viewer/4a368c163b08418890a406d413933ba7/7.52.0/en-US.

- For development systems in general, the following prerequisites apply:
  - You have an ABAP system (7.52 SP2 or higher) on an SAP HANA database version 2.0 SP01 or higher.
  - You have installed an XSA server 2.0 SP02 or higher. It is not required that XSA runs on the same system as the SAP HANA DB of your AS ABAP. You can use any XSA server that you already have (if it fulfills the release prerequisite) or install a new one with required version on a new/existing SAP HANA DB.
  - You have installed a Web IDE with a version that corresponds to your XSA version. Check the central release note for your SAP Web IDE to make sure that it fits with your XSA version (the SAP Web IDE is not needed for further configuration in this guide, but it is needed as soon as you start developing).

- For the execution of this guide, we assume that you have installed / access to the following tools:
  - SAP GUI
  - SAP HANA Studio
  - XS Command Line client: More information on the XS command line client is available on the SAP Help Portal at https://help.sap.com/viewer/4505d0bdaf4948444b7f7379d24d0f0d/p/en-US/add59469e6f444ac69cc064d131f1ee9.html
  - XS Advanced Administration Tool
  - Google Chrome is recommended for all tasks in this guide that use a browser-based UI.

1.3 Users

For this guide, you will need different users:

- One user with system administrative authorization on the ABAP side. We do not recommend that you use user DDIC, but similar authorizations are required. By using a named user, you can make sure that, later on, you can find out which user did which configurations. In this guide, the user is called MY_ABAP_USER or is referred to as ‘your user’.

  For using the report SCTS_HTA_CHECK_CONFIGURATION, the user must have the following authorizations:
  - S_TRANSPRT: ACTVT=03
  - S_OA2C_USE:PROFILE=*,ACTVT=*

  For the oAuth configuration, the user must have the following authorizations:
  - S_TCODE:TCODE=OA2C_CONFIG
  - S_OA2C_ADM:ACTVT=*
  - S_OA2C_USE:PROFILE=*,ACTVT=*

  Go to your role management in ABAP (Transaction PFCG) to add the authorizations to the role or profile that is assigned to you (the user who does the configuration) If you need more information about transaction PFCG, please take a look at the documentation in the SAP Help Portal at https://help.sap.com/viewer/b108111c22db43e6a18fab2dca108c77/1.19/en-US/4c5bd2a97817511e10000000a42189b.html ‘Role Maintenance in PFCG’

  **NOTE**
  The oAuth configuration has to be done in each client of the ABAP system that you want to use to develop HDI objects. So, you need this authorization in each client to be able to do the configuration.

- One user with administrative authorizations on the SAP HANA XSA side. We do not recommend that you use XSA_ADMIN. But similar authorizations are required. By using a named user, you can make sure that later on, you can find out which user did which configurations. In this guide, this user is called MY_XSA_USER.

- One technical user used for communication between ABAP and DI Core (DevX) / XSA. In this guide, the user is called HTA_ADMIN_AT5. It will be created in chapter Create User HTA_ADMIN_<ABAP_SID>. This user is required for the destinations 1 and 2 in the picture in chapter 1.
• (In addition, you need users in ABAP and on XSA that develop HDI objects later. In this guide, the user on XSA is called DEVELOPER. But you might need more than one developer user – one per developer. For more information about configurations required for these users, see Enable an ABAP developer for HDI Object Development.)

1.4 Further Information

• Documentation HTA for HDI on the SAP Help Portal: https://help.sap.com/viewer/6811c09434084fd1bc4f40e66913ce11/7.latest/en-US/c001a13d204a4831a068629f3934df45.html


• Installing XS Advanced Runtime: https://help.sap.com/viewer/p/SAP_HANA_PLATFORM → Installation and Upgrade → SAP HANA Server Installation and Update → Installing an SAP HANA System → Installing XS Advanced Runtime
• SAP Note 2465860 - SAP HANA 2.0 extended application services, advanced model SPS 02 release note
• SAP Note 2457320 - SAP Web IDE for SAP HANA 2.0 SPS 02 - Central Release Note
• XS command line client: https://help.sap.com/viewer/4505d0bdaf4948449b717379d24d0f0d/p/en-US/addd59069e6f444ca6ccc064d131feeec.html

2 CONFIGURATION

In this chapter, you will now execute all the configuration steps that will enable the HDI development in ABAP for your developers. See chapter Development process for details.

In some steps, you will be asked to use the report SCTS_HTA_CHECK_CONFIGURATION. This report is designed to help you with the configuration. It is not required that you use it for the configuration. Nevertheless, we recommend that you do so. The report contains checks for all configuration tasks that have to be done on the ABAP side. The checks are named like the chapters in this guide.

NOTE
We recommend that you always execute the report SCTS_HTA_CHECK_CONFIGURATION first if any issue comes up with using HTA for HDI.
If e.g. a developer reports that he cannot check out or check in any more, run the program to find out if the configuration is still ok. As a last check the report also creates some objects and deletes them again – many developer tasks are thereby included in this program and the test result can give you hints what might be wrong.

We recommend that you execute the configuration tasks in the order that is used in this guide. The report is sorted in the same way. So, checks in the report should turn into green from top to bottom if you execute the configuration tasks as described in this guide. The error messages and the corresponding long texts in the report will also offer help if the configuration did not succeed.
In addition, you can use the report if you cannot finish the configuration at once to find out where you stopped or to hand over the configuration to someone else to finish.
The report does not support or check the configuration parts that have to be done on XSA side.
When you call the report for the first time, it will look like this.
You can click on the callouts in front of each Test and error message to get more information on what is tested and what needs to be done.
The first line for each test shows the title of the chapter in this guide that is relevant for the respective test.

2.1 Checking if HDI is Enabled

HDI should have been enabled by SWPM (SAPINST) / SUM during your system setup or update for all systems that are afterwards on SAP NetWeaver 7.52. By this, the service diserver should have been started and the database schema _SYS_DI should have been created.

To be able to use HTA for HDI, a container group is needed. By default, SWPM(SAPINST) / SUM uses the same name for the container group as the ABAP schema name, e.g. SAPABAP1 or SAP<SID> in older releases.

To check if this is the case in your system, do the following:
1. Log on to your ABAP system with a user who has assigned the authorizations to use the report SCTS_HTA_CHECK_CONFIGURATION. (see chapter Users for details).

2. Open transaction SA38
3. Enter the Program SCTS_HTA_CHECK_CONFIGURATION and Execute it

4. The first check Test: ‘Checking if HDI is enabled’ should be green. (You can open the details for Check if HDI is enabled’ to get more details about what it means that HDI is enabled)

   ![ABAP: Program Execution](image)

If this is not the case, check SAP Note 2603193 Enable HTA for HDI on ABAP systems after the installation with SWPM. If this doesn’t solve the issue, open an incident for SAP in the component BC-INS-SWPM, if you newly installed the system, and in component BC-UPG-TLS-TLA, if the system was upgraded.

2.2 Preparing the Communication from XSA to ABAP

You need to create SAP HANA procedures to be able to create technical users for the External Services (User-Provided Services) that are required for the communication from Web IDE on XSA to the HANA DB of AS ABAP – even if they run on one system. The technical users will be created automatically later, as soon as developers start developing HDI artifacts. This is illustrated by the destination in the picture above. The technical users that will be created (later on) with the help of the SAP HANA procedures need a dedicated role which you will create on the SAP HANA that is used by ABAP (not on the SAP HANA of the XSA system if it is a different one).

You will perform both actions by executing an SQL statement on the tenant database of the SAP HANA system that is used by ABAP.

2.2.1 Creating Database Procedures Required by AS ABAP

To create the required SAP HANA procedures, you have to execute SQL statements on the relevant tenant database of your SAP HANA system which is used by ABAP. Before execution, adapt the script with your own data by following the instructions at the beginning of the script.

**NOTE**

In our example, this needs to be done on HDB (which is the SAP HANA DB underlying the ABAP system), and not on ZDM (which is the database for XSA).
1. Execute the report SCTS_HTA_CHECK_CONFIGURATION in transaction SA38. The test Test 'Creating Database procedures required by AS ABAP' should show an error.

![Test of HDI configuration](image1)

2. Open the SAP HANA Studio and log on to the tenant DB and not the system DB of your SAP HANA system underlying ABAP.

![SAP HANA Studio](image2)

3. Right-click the tenant database and choose Open SQL Console.

![Open SQL Console](image3)

4. Download the file CreatingDatabaseProceduresRequiredByASABAP.SQL from SAP Note 2569651 and save it to your PC or copy the code from the Appendix: Script for Creating Database Procedures Required by AS ABAP.

5. Load the script into the SQL console – right-click in the empty space of the SQL console and choose Open File… and locate the file that you saved in the previous step or copy the code into the SQL.
6. Adapt the script according to the instructions at the beginning of the script:
   a. Set the correct password for user SYSTEM by replacing `<manager>` with the password of the SYSTEM user.
   b. Set the correct name for the user SAPSID by replacing `<USER>`. The SAPSID user name is usually SAPABAP1. You can check the user name in your ABAP system using System ➔ Status. In the Database Data area, it is displayed as value for the User field.
   c. Set the correct name for the ABAP schema by replacing `<SCHEMA>` with the name of the ABAP schema in this system. You can check the schema name in your ABAP system using System ➔ Status. In the Database Data area, it is displayed as value for the Schema field.
NOTE
If you used values other than standard SAPABAP1 for the Schema and the User when setting up your system, you have to configure those values by the help of the report SCTS_HTA_CONFIGURATION. Check the chapter Using the Report SCTS_HTA_CONFIGURATION for details.

7. Select all lines of the script (CTRL+A) and execute them at once.

8. Make sure that no errors are reported. If errors are reported which are not related to the replacements that you did, check SAP Note 2569651 Configure your ABAP Development System for Development of HDI Objects.

9. Execute the report SCTS_HTA_CHECK_CONFIGURATION in transaction SA38. The test Test 'Creating Database procedures required by AS ABAP' should now be ok.

2.2.2 Creating an SAP HANA Role
The technical users that will be created (later on) with the help of the procedures of the previous chapter need a dedicated role. In this chapter, you are going to create this role on the SAP HANA system that is used by ABAP (not on the XSA system if it is a different one).
To do this, you proceed as above (download the file with the SQL statements and open it in an SQL console of the tenant database of your SAP HANA system and execute it after having adapted it with your own data by following the instructions at the beginning of the script).
NOTE

In our example, this needs to be done on HDB (which is the SAP HANA DB underlying the ABAP system) and not on ZDM (which is the database for XSA).

1. Execute the report SCTS_HTA_CHECK_CONFIGURATION in transaction SA 38. The test Test ‘Creating an SAP HANA Role’ should show an error (Role ‘HTA_UPS_COMMON’ does not exist).

2. Use an empty SQL-Console (Right-click the tenant database and choose Open SQL Console to open a new one):

3. Download the file CreatingAnSAPHANARole.sql from SAP Note 2569651 and save it to your PC or copy the code from the Appendix: Script for creating SAP HANA role.

4. Load the script into the SQL console – right-click in the empty space of the SQL console and choose Open File… and locate the file that you saved in the previous step or copy the code into the SQL console.
5. Adapt the script according to the instructions at the beginning of the script: Set the correct password for user SYSTEM by replacing `<manager>` with the password of the SYSTEM user in the relevant ABAP system.

6. Select all (CTRL+A) and execute the script.

If errors are reported for GRANT statements, you can ignore them. Depending on the HANA version there are more/less views/table/procedures available. For other errors, or if you are in doubt, create a ticket and assign it to component BC-CTS-HTA.

7. Execute the report SCTS_HTA_CHECK_CONFIGURATION in transaction SA38. The test Test ‘Creating an SAP HANA Role’ should now be ok.

2.3 Preparing the Communication from ABAP to XSA

For the communication from ABAP to DevX / XSA a technical user is needed on XSA with dedicated roles. We will now at first create the required role collections. After that, we will create the user (see chapter Create User HTA_ADMIN <ABAP_SID>) and assign the role collections to the user.

2.3.1 Creating Role Collections in XSA

**NOTE**

This configuration step has to be executed on XSA – the program SCTS_HTA_CHECK_CONFIGURATION can therefore not be used to test whether the configuration is already done and is working fine.

1. Open XS Advanced Administration (Admin UI). The URL looks like this: https://<server>:<port of app>/index.html. To find the correct port, you can use the xs command line tool of your XSA installation.

2. First log on to the server with an OS user.
   Call the command `xs login`
   Log on with an XSA Admin user
3. Use the command `xs-v`.

Look for the Registered service URLs for `xsa-admin` and copy the URL to a browser window.

(If you don’t know this command line tool or the URL, ask your XSA administrator.)
4. Log in with an admin user.

5. Choose the Application Role Builder tile.

If you cannot see this tile, ask your administrator for permissions.

6. Choose Role Collection from the left navigation bar.
7. Choose the plus sign at the bottom of the screen to create a new role collection.

If the role collections already exist in your system, please check whether they contain the correct roles.

NOTE
If you already have your own role collections you do not have to create new role collections, but you can assign the application roles mentioned in the following steps to your own role collections.
More information about assigning SAP Web IDE role templates to role collections is available on the SAP Help Portal at https://help.sap.com/viewer/p/SAP_HANA_PLATFORM → Installation and Upgrade → SAP Web IDE for SAP HANA - Installation and Upgrade Guide (Click on View All in the section Installation and upgrade if you cannot see the link for the Web IDE guide immediately) → Post-Installation Administration Tasks → Enabling Access to the SAP Web IDE Administration and Development Tools and in a short video: https://video.sap.com/media/t/1_9lfjcp1v

8. Enter a Name and a Description and choose Create.
9. Select the new role collection and choose the Roles tab.

10. Choose Add Application Role.

11. From the dropdown menus in the Add Application Role dialog box, choose the following entries:

<table>
<thead>
<tr>
<th>Application Name</th>
<th>Role Template</th>
<th>Role Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>webide11</td>
<td>WebIDE_Developer</td>
<td>WebIDE_Developer</td>
</tr>
<tr>
<td>Template Name</td>
<td>WebIDE_Developer</td>
<td></td>
</tr>
<tr>
<td>Application role</td>
<td>WebIDE_Developer</td>
<td></td>
</tr>
</tbody>
</table>
You have to do this top down – the entries in the second and third drop-down will only appear after something was selected for the first and second drop-down. Choose Ok.

12. At the bottom of the screen, choose Save.

13. Repeat steps 3-10 to create the role collection WEBIDE_ADMINISTRATOR. Use the following values as Name and Description:

<table>
<thead>
<tr>
<th>Role Collection Name</th>
<th>WEBIDE_ADMINISTRATOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Role Collection Description</td>
<td>WEBIDE_ADMINISTRATOR</td>
</tr>
</tbody>
</table>

On the Roles tab, choose the following values

<table>
<thead>
<tr>
<th>Application Name</th>
<th>webide\i1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Template Name</td>
<td>WebIDE_Administrator</td>
</tr>
<tr>
<td>Application Role</td>
<td>WebIDE_Administrator</td>
</tr>
</tbody>
</table>

The result looks like this:

WEBIDE_ADMINISTRATOR

1
Details: Roles

<table>
<thead>
<tr>
<th>Application Name</th>
<th>Role Template</th>
<th>Role Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>webide\i1</td>
<td>WebIDE_Administrator</td>
<td>WebIDE_Administrator</td>
</tr>
</tbody>
</table>
2.3.2 Create User HTA_ADMIN_<ABAP_SID> in XSA

NOTE
This configuration step has to be executed on XSA – the program SCTS_HTA_CHECK_CONFIGURATION can therefore not be used to test whether the configuration is already done and is working fine.

1. Use the Home button to go back to the start page of the XS Advanced Administration and Monitoring Tools (Admin UI). (If you closed the UI, check Creating Role Collections in XSA to find out how to open this UI.)

2. Choose the User Management tile.
3. Choose New.

4. Fill in the required data for the user HTA_ADMIN_<ABAP_SID> (HTA_ADMIN_AT5, in our example, as this user is used on the ABAP server to communicate with DevX / XSA).

   Note that the password that you enter here, is an initial password that you will need to change a later step. Therefore, use a generic password, such as Init1234. Choose Create.
5. Click on the arrow behind the user’s data.

6. Choose the Role Collections tab.

7. Choose Add.

8. In the Role Collections dialog box, select the WEBIDE_DEVELOPER and the WEBIDE_ADMINISTRATOR role collections that you created before, and XS_CONTROLLER_USER. Choose OK.
The three role collections were assigned to the user. The role \texttt{XS\_USER\_PUBLIC} is assigned to each user automatically.

9. Choose Save.
10. Since this user is a technical user that is only used in the background, log on once with this user to the \textit{SAP HANA XS Advanced} to change the initial password. You will need the new password in a later step so make sure that you remember it or note it down in a password safe-tool.
You can ignore messages like *Error in fetching applications* when you log on with the user
HTA_ADMIN_AT5

11. Assign the role collection *WEBIDE_ADMINISTRATOR* also to your user (user My_XSA_User in our example). You will need this authorization later to log on to the DI Space Enablement UI (see Enable HDI in new space).

### 2.3.3 Creating Destinations in ABAP

ABAP is the leading system when developing HDI objects for ABAP on HANA applications. Therefore, ABAP needs to be able to connect to XSA / DevX and ABAP needs to configure XSA to connect back to the SAP HANA database of AS ABAP. In detail, the destination XSA_ADMI is needed by ABAP to create user provided services and XSA_DEVX is required to create a shared workspace, add users to this workspace, create projects, and copy objects from the shared workspace to ABAP and vice versa.

In this chapter, we will use a standard naming for these two destinations. If you stick to this naming, no further action is required. If you choose other names, you have to make the system, aware of these by entering the destination names in the report SCTS_HTA_CONFIGURATION. Check the chapter Using the Report SCTS_HTA_CONFIGURATION for details.

1. Go to transaction SA38 and execute the program SCTS_HTA_CHECK_CONFIGURATION.
The test Test: ‘Creating Destinations in ABAP’ should show an error.
2. Go to transaction SM59.

3. Click on Create to create a new destination.

4. Enter XSA_ADMI as RFC Destination. Enter Connection Type G and e.g. Destination to XSA Controller as Description 1.

Press <enter>. 
5. Confirm the pop-up

6. Execute the command `xs -v` to get host and port of the `controllerEndpoint`.

7. Enter the values as `Targ` and `Serv` on the tab `Technical Settings` of your destination `XSA_ADMI`. Make sure that you enter the values that you got as output when executing the command `xs -v`. The values for your system will differ from the ones shown in this guide.
Usually, the port (so what you enter for Serv) looks like the following: 3<XSA Instance>30

   Set SSL to Active.
   Confirm the pop-up that user and password are going to be deleted.

   Choose **ANONYM SSL Client (Anonymous)** from the drop-down for **SSL Certificate**.
If you get a warning *SSL client PSE ANONYM does not exist*, see troubleshooting section Cannot select ANONYM SSL Client (Anonymous) from the drop-down for SSL Certificate.

If you can't select the *SSL Certificate ANONYM SSL Client (Anonymous)*, the SSL-client configuration was not yet done for this ABAP system. See troubleshooting section Cannot select ANONYM SSL Client (Anonymous) from the drop-down for SSL Certificate.

9. Save your settings.

10. Execute the *Connection Test*. 
11. The value for *Status HTTP Response* should be 200.

![Connection Test HTTP Destination XSA_ADMI](image)

If the result is not 200 or if you receive a pop-up showing an error message check the troubleshooting section *Error when executing connection test for destination*.

12. You need a second destination named XSA_DEVX. To create it, copy the destination XSA_ADMI that you just created.

![Copy Command](image)

13. Enter XSA_DEVX as *Destination* and click *OK*.

![Destination XSA_DEVX](image)

14. Switch to Edit mode.
15. To get the URL and the port, execute the command `xs target -s SAP` first to make sure that you work in the SAP space and then execute the command `xs app di-core`.

16. Enter a description. Check the host, enter the port and save the destination. (Leave all other settings unchanged.)

17. Execute the connection test for this destination. The following dialog popup will appear.

On this dialog popup choose `Cancel`. You will get the following window (which is no error in this case).

If the above dialog popup does not appear, see chapter Error when executing connection test for destination.
18. Go to transaction SA38 and execute the report SCTS_HTA_CHECK_CONFIGURATION. The test Test: ‘Creating Destinations in ABAP’ should now be ok.

2.3.4 Configuring the Communication from ABAP to XSA

Since the communication between AS ABAP and SAP HANA XS advanced model requires the OAuth 2.0 protocol, we will configure this communication between AS ABAP and the backend of SAP Web IDE (DevX) and AS ABAP and the XS controller. This is done by maintaining the OAuth2 client profiles SAP_HTA_HDI and SAP_HTA_HDI_XSA. This configuration has to be done in each client that you use for developing HDI objects.

1. Go to transaction SA38 and execute the report SCTS_HTA_CHECK_CONFIGURATION. The test Test: ‘Configuring the Communication from ABAP to XSA’ should show errors.

2. In the previous chapter, you used the XS command line client. If the window is still open, continue using it (make sure that you are working in the space SAP). If you closed the window, check the previous chapter to find out how to log in and to switch to the space SAP).

3. Execute xs create-service-key devx-uaa devx-uaa-ABAP.

Both, OK or failed: Service key with name "devx-uaa-ABAP" already exists are fine as results for this step.

4. Execute xs sk devx-uaa devx-uaa-ABAP.

Copy the values for clientid and clientsecret and paste them to e.g. notepad because you will need it
for subsequent steps.

5. Go to your SAP GUI for system AT5. Open transaction OA2C_CONFIG.

NOTE
OAuth Clients created in transaction OA2C_CONFIG are specific to the ABAP client that you are logged on to. So, make sure that you do this configuration in the correct ABAP client (the one that you will use for developing HDI objects – the same that you use also for developing ABAP workbench objects). If you use more than one client of your ABAP system for developing HDI objects, you have to do this configuration in OA2C_CONFIG in each client.

NOTE
If a new browser window opens up showing Forbidden and Service cannot be reached as title, check the chapter Enable transaction OA2C_CONFIG
6. Click on Create.

7. Choose SAP_HTA_HDI as OAuth 2.0 Client Profile and the OAuth 2.0 Client ID that you found in step 4.

8. Fill in the details in the order listed here – make especially sure that you fill in the Client Secret at last.
   a. Authorization Endpoint: Execute the command `xs -v` to get the URL.

   **CAUTION:** Make sure that you enter the URL without ‘https://’. Add `/oauth/authorize` at the end.
b. **Token Endpoint**: Use the same URL as for Authorization Endpoint but replace `authorize` at the end by `token`.

c. **Access Settings**: Set **Client Authentication** to **Basic**, set **Resource Access Authentication** to **Header Field** and **Selected Grant Type** to **Resource Owner Password Credentials**.

d. **Client Secret**: Enter the value that you found in step 4.
9. Click Save.

10. As a result, you should see a message that OAuth 2.0 Client sb-webide11 is saved successfully. The list of existing OAuth 2.0 Clients should now contain the client that you just created.

11. Scroll down until you can see the button Request Tokens… Click on it.

   (If the client is shown in display mode, click on Edit, first.)

12. Enter your user HTA_ADMIN_<ABAP_SID> (HTA_ADMIN_AT5 in our example) and the user’s password. Click OK.

   If you get an error when clicking OK, see Error when requesting OAuth2.0 Tokens.

13. As a result, the circle in front of Request Tokens should change from red to green. Now the communication between ABAP and XSA is established.

   CAUTION: the token is only valid for 30 days. You have therefore to request tokens every thirty days. If the token has expired, the developer will see the error message ‘Refresh Token has Expired at the Server (Execute OAuth 2.0 Access Token Request Again).’ In the HDI namespace editor in ADT. Check Guided Answers for details: https://gad5158842f.us2.hana.ondemand.com/dtp/viewer/#/tree/2311/actions/30100:30571:30106/?version=current
14. Go to transaction SA38 and execute the report SCTS_HTA_CHECK_CONFIGURATION. The test Test: ‘Configuring the Communication from ABAP to XSA’ should now be ok.

2.4 Configuring the Space

More information on creating and managing spaces is also available on the SAP Help Portal at https://help.sap.com/viewer/1a8e7ab05a2e4119b026b702121142215/2.0.02/en-US/a6a17b3b99d4280b5e5c354d0d74dab.html ‘Managing Spaces for Development’

2.4.1 Creating XSA Space ABAP_SAPAT5

In this chapter, you are going to create a space on your XSA system that is used by HDI. The default naming of the space is ABAP_SAP<ABAP-SID>. AT5 is the SID of the ABAP system that we use as an example in this guide. When you do the configuration on your system, you will see the SID of your ABAP system.

1. Go to transaction SA38 and execute the report SCTS_HTA_CHECK_CONFIGURATION. The test Test: ‘Creating XSA Space ABAP_SAP<SID>’ should show errors.

2. Go back to the home page of XS Advanced Administration and Monitoring Tools (Admin UI). (If you closed the UI, check the beginning of the chapter Creating Role Collections in XSA to find out how to open this UI.)
3. Choose the Organization and Space Management tile.

4. Choose Create Space.

5. In the Create Space dialog box, enter ABAP_SAP<ABAP_SID>, and choose Create.
We recommend that you use this standard naming. If you do not use this standard space name ABAP_SAP<ABAP_SID> in here, you have to add the space name to the configuration of HTA for HDI in the report SCTS_HTA_CONFIGURATION. See Using the Report SCTS_HTA_CONFIGURATION

6. Click Save.

7. Select the new space and choose Users. Choose Add User.
8. Select HTA_ADMIN.<ABAP_SID> from the list, or search for it and choose OK.

9. Select Developer for this user and choose Save.

10. Repeat the steps 7 to 9 for your user (user My_XSA_User in our example). This is required because you will use this user to enable development in this space.

11. You can assign additional users to this space according to your needs. You can also do so later on at any point in time.
12. Go to transaction SA38 and execute the report SCTS_HTA_CHECK_CONFIGURATION. The test Test: ‘Creating XSA Space ABAP_SAP<SID>’ should now be ok.

![Test of HDI configuration](image1)

2.4.2 **Enable Development in Space ABAP_SAP<ABAP_SID>**

**NOTE**

This configuration step has to be executed on XSA – the program SCTS_HTA_CHECK_CONFIGURATION can therefore not be used to test whether the configuration is already done and is working fine.

1. Open the command line tool (see beginning of chapter [Create Web IDE Roles](#) for details). Use the command `xs -v`. Look for the URL `di-space-enablement-ui` and copy it.

[Registered service URLs](image2)

(If you don’t know this tool or the URL, ask your administrator.)

2. Open a new browser window and enter the URL that you just copied. Log on with the user `MY_XSA_USER`.

![SAP HANA XS Advanced](image3)

If you get an error message when logging on to the DI Space Enablement UI, your user may not have the appropriate authorization. See troubleshooting section [Space enablement UI does not open, error message: Forbidden](#).
3. You should see just the space that you created before. Click **Enable** for this space.

![Enable space](image)

4. Wait until you can see a message *The builder has been deployed successfully*. It might be that you have to scroll down in the **Log** area.

![Space Enablement](image)

5. Come back to this UI after each update of Web IDE or DevX and check whether you have to update the space: if the status for the space is **Outdated**, execute an **Update**

![Update space](image)

### 2.5 Maintaining the Web IDE URL on ABAP

You need to define the URL to your Web IDE to allow your developers later on to easily navigate to Web IDE: the URL is used in ADT to enable a button named *Open in Web IDE* in the namespace editor. With this, the developers can easily reach the right namespace in the Web IDE from ABAP development tools.
1. Go to transaction SA38 and execute the report SCTS_HTA_CHECK_CONFIGURATION. The test Test: 'Maintaining the Web IDE URL on ABAP' should show an error.

2. We assume that you use the Web IDE on the system where XSA is installed. Execute the command `xs app webide` on your system where XSA is installed (ZDM in our example) and copy the value for `urls`.

   If you get the error message `xs app webide not found in space <xyz>`, you may have to switch first to the space SAP before you can execute the command `xs app webide`. To switch to the space SAP, execute the command `xs target -s SAP`.

3. Go back to transaction SA38.

4. Execute the program SCTS_HTA_CONFIGURATION.

5. The report shows the standard values for all parameters except for the Web IDE URL. As URL of SAP Web IDE for SAP HANA enter the `urls` value that you copied in step 2. Leave all other input fields as they are. The default values are shown – you only need to change them if you did not follow
the standard configuration and naming as described in this guide. Details about when else to use this report are described in chapter Using the Report SCTS_HTA_CONFIGURATION.

6. If you did not use standard naming in the previous configuration steps, you have to change the values for the respective parameters as well. (For Name of XSA space, see Creating XSA Space ABAP_SAP<ABAP_SID>, for destinations, see Creating Destinations in ABAP. CreateUserProcedure is part of the SQLScript that you executed in Creating Database Procedures Required by AS ABAP.)

7. Execute the report after you filled in the data that you need.

8. You should see a result similar to the following (depending on the data that you maintained in the previous steps). For each parameter, you can see the current value and the default value.
9. Go to transaction SA38 and execute the report SCTS_HTA_CHECK_CONFIGURATION. The test: Test: 'Maintaining the Web IDE URL on ABAP' should now be ok.

2.6 Creating the Shared Workspace

When working with HTA for HDI, all developers use a shared workspace. With this, they can work on the same objects. They don’t use any local workspace when working in SAP Web IDE for SAP HANA.

1. Go to transaction SA38 and execute the program SCTS_HTA_CHECK_CONFIGURATION. The test Test: 'Creating the shared Workspace' should show an error.
2. Go back and execute the report `SCTS_CTS_HTA_HDI_WSM_CONFIGURE`.

3. Choose `Create workspace / get existing workspace` as `Action`. Empty the field `User ID` if a value is shown when you start the report. Click on `Execute`.

4. The workspace is created, and the `Workspace id` is shown. Click `OK`.

If you do not get the workspace shown but get errors, see Error when creating shared Workspace.

5. Go back again and execute the report `SCTS_HTA_CHECK_CONFIGURATION`. This was the last configuration step. The program `SCTS_HTA_CHECK_CONFIGURATION` will now also execute a complete functional test for HTA for HDI. Some objects will be created in the background and will gain be deleted. Therefore, it will take some time until the program is executed, this time. The test `Test: 'Creating the shared Workspace'` should now be ok.

6. As this was the last configuration step, the report now executes the test `Checking the configuration` as well. This test creates several objects and deletes them again. Therefore, the execution of the
program takes longer this time.

![Testing of HDI configuration](image)

7. The test Test ‘Checking the Configuration’ should afterwards also be ok – you can open the details to find out what had been tested including which objects had been created for testing purposes and had been deleted afterwards again. You can also check the configuration for yourself – the steps are described in the chapter Check the Configuration (optional).

### 2.7 Enable an ABAP Developer for HDI Object Development

Each developer user on ABAP must have a corresponding user on XSA. A developer user on ABAP needs the authorization S_OA2C_USE:PROFILE=;ACTVT=.

The XSA developer user must be assigned as developer to the space that you created before, he has to have Web IDE developer rights (using the role collection that you created before), and he must have authorizations on the Web IDE shared workspace. It is not required, that a developer has the same user-ID on both sides (ABAP and XSA). As it is not possible to set up SSO between HTA for HDI and XSA, this has to be done in any case for every user who shall have developer permissions in XSA.

#### 2.7.1 Assigning Authorizations to the Developer on ABAP

A new authorization object enables your developers to use the destinations that you just configured. Go to your role management in ABAP (Transaction PFCG) to add the authorization S_OA2C_USE:PROFILE=;ACTVT= to the role or profile that is assigned to your developers.

If you need more information about transaction PFCG, please take a look at the documentation in the SAP Help Portal at “Role Maintenance in PFCG” [here](https://help.sap.com/viewer/9737050ef01843f19572591b42128f1b/7.1.0.0000000a42189b.html).

#### 2.7.2 Creating a Developer User on XSA

If accounts for your developers already exist on XSA, continue with chapter Assigning Developer Role in XSA.

To create a new developer user on XSA, proceed as follows:
1. Open the User Management in XS Advanced Administration on the system where XSA is installed (in our example, this is system ZDM).

2. Click on New to create a new user.

3. Enter all required fields in the dialog box. In our example, the user is called DEVELOPER. Afterwards, choose Create.

**NOTE**
If the user already exists in SAP HANA, you can click on the tab *Migrate User* to transform this user into an XSA user.

The DEVELOPER user was created.

### 2.7.3 Assigning Developer Role in XSA

Each user who shall develop has to have the role *WEBIDE_DEVELOPER* assigned that you created in chapter *Creating Role Collections in XSA*.

1. You are in the *User Management* in *XS Advanced Administration* on the system where XSA is installed (in our example, this is system ZDM).
2. Open the details for your user.
3. Click on *Role Collections*.
4. Select ‘Add’ and select the role collection **WEBIDE_DEVELOPER** and click **OK**.

5. Click on **Save**.

---

**2.7.4 Adding Developer User to Space**

Each developer has to be assigned to the correct space.
1. Open Organization and Space Management

2. Open the details for your space (ABAP_SAPAT5 in our example)

3. Click on Users
4. Click on Add User

5. Mark your user and click OK
6. Make the user *Developer* and click on *Save*

7. Repeat these steps for any user who will develop. You do not have to do this for all users right now. Just re-execute the steps described in this chapter whenever you need to add a user.

### 2.7.5 Assigning the XSA User to the Web IDE Shared Workspace

You can repeat this step whenever a new developer user needs to develop HDI objects and thus needs to work on the shared workspace. It is not required to assign all the possible developers right now.

1. Open transaction SA38.

2. Execute the report `SCTS_CTS_HTA_HDI_WSM_CONFIGURE`.

3. Choose the *Action Add user with collaborator right*. Enter the *User ID* of your developer on XSA (DEVELOPER in our example) and click *Execute*.
3 CHECKING THE CONFIGURATION (OPTIONAL)

The program SCTS_HTA_CHECK_CONFIGURATION checks for you whether your configuration is complete and works fine. After you created the shared workspace, the complete configuration check is also executed.

Test of HDI configuration

The test Test: 'Checking the Configuration' should also be green – you can open the details to find out what had been tested including which objects had been created for testing purposes and had been deleted afterwards again. You can also check the configuration for yourself. A tutorial on the SAP Help Portal guides you through the steps for creating some basic HDI objects: Tutorial: Developing and Consuming HDI Objects in ABAP.

4 TROUBLESHOOTING

In this chapter, you can find possible errors and their solutions. The title of the sub-chapter is the error message that you can see – or the most 'speaking' part of it. This chapter contains issues that could come up when you configure HTA for HDI.

If you encounter issues during development, please refer SAP Note 2682272.
1. NOTE

We recommend that you always execute the report SCTS_HTA_CHECK_CONFIGURATION first if any issue comes up with using HTA for HDI.
If e.g. a developer reports that he cannot check out or check in any more, run the program to find out if the configuration is still ok. As a last check the report also creates some objects and deletes them again – many developer tasks are thereby included in this program and the test result can give you hints what might be wrong.

4.1 Cannot Select ANONYM SSL Client (Anonymous) from the Drop-Down for SSL Certificate

ISSUE: you get a warning SSL Client PSE Default is not available or you are not able to select ANONYM SSL Client (Anonymous) from the drop-down for SSL Certificate when you create the destinations in chapter Creating Destinations in ABAP.
SOLUTION: check transaction STRUST and make sure that the section SSL client SSL Client (Anonymous) is created (shown as a folder) and certificates are ok. See Appendix Configuring Secure Connection from ABAP to XSA for details.

4.2 Error when Executing Connection Test for Destination

4.2.1 HTTPIO_PLG_CANCELED

ISSUE: when you execute the connection test for the destinations created in chapter Create Destinations in ABAP You get the message HTTPIO_PLG_CANCELED
SOLUTION: Check whether http(s)-services are started correctly:
Open transaction SMICM

Choose Goto → Services.

Check the both HTTP and HTTPS-Protocol are marked as Active.
4.2.2 **SSL Handshake Fails**

ISSUE: When checking the destination XSA_ADMI in chapter *Creating Destinations in ABAP* you get the following error message:

![SSL handshake failure](image)

SOLUTION: Make sure that SSL is set up correctly. For details have a look at chapter *Configuring Secure Connection from ABAP to XSA*.

4.2.3 **Connection to <host>.<port> failed: NIEHOST_UNKNOWN(-2)**

ISSUE: When executing the connection test for destination XSA_DEVX in chapter *Creating Destinations in ABAP* you get the following error message:

![Connection test failure](image)

SOLUTION: Make sure that the target host and port number is set correctly. Check especially that you did not copy `https://` into the `Targ`-field.

4.2.4 **Connect to <host>:<port> failed: NIECONN_REFUSED(-10)**

ISSUE: When executing the connection test for destination XSA_DEVX in *Creating Destinations in ABAP* you get the following error message:

![Connection test failure](image)

SOLUTION: Make sure that the target host and port number is set correctly.
4.2.5 **No Logon-pop-up**

ISSUE: when you execute the connection test for the destination XSA_DEVX, there is no log-on pop-up shown. The *Status HTTP Response* is 404 (and not 401)

![Connection Test HTTP Destination XSA_DEVX](image)

SOLUTION: Check that you entered the correct port

4.2.6 **Communication from ABAP to XSA failed**

ISSUE: after you configured the destination XSA_DEVX, the program SCTS_HTA_CHECK_CONFIGURATION shows the error message *Communication from ABAP to XSA failed* for the test *Configuring the Communication from ABAP to XSA.*

![RFC Destination XSA_DEVX](image)

SOLUTION: execute the command `xs app di-core` and make sure that the port shown is the one that you entered in the destination XSA_DEVX in SM59. It can happen that the port changes if you re-installed the Web IDE.

ISSUE after having configured the *communication from ABAP to XSA*, you get the following error message when executing the report SCTS_HTA_CHECK_CONFIGURATION:
SOLUTION: (only if you are on Windows) implement SAP Note 2677819.

4.3 Enable transaction OA2C_CONFIG

ISSUE: When you try to open transaction OA2C_CONFIG, you receive the error message Forbidden in the browser window (with the title Service cannot be reached)

SOLUTION: Check whether the required service is activated

Log on to your ABAP system and open transaction SICF. Check that the service OA2C_CONFIG is activated. If it is not activated, yet, activate it.
4.4 Error when Requesting OAuth 2.0 Tokens

4.4.1 HTTP Failure, Processing Failed, Invalid State, Invalid Timeout or Others.

ISSUE:

SOLUTION: check that you entered the URLs for the endpoints correctly. Especially, make sure that you did not enter http://, https:// or spaces at the beginning of the URL and that you used correct port of the UAA of your XSA installation. Especially check what is marked in the following screenshot:
4.4.2  **Client Authentication Failed (e.g., Unknown Client, no Client Authentication Included, …)**

**Issue:** You get the following error popup when you execute Request Token for an OAuth client.

![Image of Request OAuth 2.0 Tokens popup]

**Solution:** This error is either related to a wrong password or Client Secret. First try to execute Request Token again and make sure to use correct password. If this still fails, edit the OAuth 2 client configuration by reentering the Client Secret. Then save the configuration and retry Request Token.

4.4.3  **The Server is Refusing to Respond to the Request (HTTP 403 – Forbidden)**

**ISSUE:** You get the following error popup when you execute Request Token for an OAuth client.

![Image of Request OAuth 2.0 Tokens popup]

**SOLUTION:** Check whether the user HTA_ADMIN_AT5 needs to change his password by trying to logon to XS Advanced Administration and Monitoring Tools (Admin UI) and change the password there. Then retry to request OAuth 2.0 Tokens in transaction OA2C_CONFIG. This has to be done in each client where you develop.

4.4.4  **Requested Resource could not be found (HTTP 404 – not found)**

**ISSUE:** You get the following error popup when you execute Request Token for an OAuth client

**SOLUTION:** Check the URLs for Authorization Endpoint and Token Endpoint. Make sure that they follow the following format:

```
<server>:<port>/uaa-security/oauth/authorize
<server>:<port>/uaa-security/oauth/token
```

4.5  **Error in existence check for XSA space**

4.5.1  **Error when executing GET_SPACE_ID command (http code is 0)**

**ISSUE:** after having configured what is described in chapter Creating XSA Space ABAP_SAPAT5, the report SCTS_HTA_CHECK_CONFIGURATION shows the error message "Error when executing GET_SPACE_ID command"
4.6 Space Enablement UI Does not Open, Error Message: Forbidden

ISSUE: When you try to open the Space Enablement UI, you get the error message ‘Forbidden’

SOLUTION: To log on to the DI Space Enablement UI, the logon user needs the WEBIDE_ADMINISTRATOR authorization. If you get the error message Forbidden, assign the WEBIDE_ADMINISTRATOR authorization to the logon user.

4.7 Error when Creating Shared Workspace

When creating the shared workspace (Chapter Creating the Shared Workspace) you get a sequence of Information Popups with error messages. Usually the first popup looks like this:

The following popups depend on the error situation and are shown in the following sub chapter.

4.7.1 Action CREATE HTTP REST CLIENT was canceled (http status code 0)

ISSUE: When creating the shared workspace, you get the following popup:

SOLUTION: Check that you entered the correct host name and port number for destination XSA_DEVX. Make sure you used the port that is shown when you execute the command xs app di-core on your system. Do not use the port shown in this guide. Ports differ from system to system. See chapter Creating Destinations in ABAP, especially steps 15 and 16.
4.7.2  **Action CREATE WORKSPACE(POST) was Canceled (Http Status Code 404)**

**ISSUE:** When creating the shared workspace, you get the following popup:

![](image)

**SOLUTION:** Check that you entered the correct host name and port number for destination XSA_DEVX. Make sure you used the port that is shown when you execute the command `xs app di-core` on your system. Do not use the port shown in this guide. Ports differ from system to system. See chapter [Creating Destinations in ABAP](#), especially steps 15 and 16.

4.7.3  **Action CREATE WORKSPACE(POST) was Canceled (Http Status Code 500)**

**ISSUE:** When creating the shared workspace, you get the following popup:

![](image)

**SOLUTION:** Check the OAuth Client configurations whether all settings are correct. As first step open transaction OA2C_CONFIG, in the table select the line with `Configuration Name = SAP_HTA_HDI` and `OAuth 2.0 Client ID = sb-webide/!1` and check that the button `Request Token` shows a green circle. If this is not the case switch to edit mode and press `Request Token` and provide username as `HTA_ADMIN_<ABAP_SID>` with the correct password. See also chapter [Configuring the Communication from ABAP to XSA](#).

4.7.4  **Invalid action**

**ISSUE:**

![](image)

**SOLUTION:** Check that you entered the correct host name and port number for destination XSA_DEVX.
4.7.5 OAuth 2.0 Client Profile SAP_HTA_HDI is not Assigned to an OAuth 2.0 Client.

ISSUE: When creating the shared workspace, you get the following popup:

![OAuth 2.0 Client Profile SAP_HTA_HDI is not assigned to an OAuth 2.0 Client.]

SOLUTION: Check the OAuth Client configurations whether the required clients were created. In SAP GUI start transaction OA2C_CONFIG and make sure you find the OAuth 2.0 Client IDs, sb-webide11 and sb-admin.

If one or the other is missing, see chapter Configuring the Communication from ABAP to XSA and create them.

5 APPENDIX

5.1 Script for Creating Database Procedures Required by AS ABAP

-- How to Use this script!
-- 0. If this is a MDC/MultiDB HANA system, connect to the tenant DB and NOT to the SYSTEMDB
-- 1. Set correct password for user SYSTEM by replacing all occurrences of <manager>!!!
-- 2. Get the user name with SAP GUI -> System -> Status... -> Database Data - User. Copy the value for User. Replace all occurrences of <USER> with the value that you copied
-- 3. Get the schema name for your system: in SAP GUI, go to System -> Status... -> Database Data - Schema. Copy the value for Schema. Replace all occurrences of <SCHEMA> with the value that you copied.
-- 4. Select all statements and execute them in one shot
-- 5. Make sure there were no errors reported during execution of the script. If errors were reported, created ticket on BC-CTS-HTA

-- login with a USER ADMIN user
CONNECT SYSTEM PASSWORD <manager>;

-- create a technical user as the owner of the
CREATE TECHNICAL_USER/DROP_TECHNICAL_USER procedures
CREATE RESTRICTED USER <SCHEMA>_HTA_USER_ADMIN PASSWORD abcd1234ABC_D NO
FORCE_FIRST_PASSWORD_CHANGE;

-- add CREATE ANY on own schema to the technical user
ALTER USER <SCHEMA>_HTA_USER_ADMIN GRANT CREATE ANY ON OWN SCHEMA;

-- allow this user to connect
ALTER USER <SCHEMA>_HTA_USER_ADMIN ENABLE CLIENT CONNECT;
--GRANT RESTRICTED_USER_JDBC_ACCESS TO <SCHEMA>_HTA_USER_ADMIN; -- granted below with grant option
--GRANT RESTRICTED_USER_ODBC_ACCESS TO <SCHEMA>_HTA_USER_ADMIN; -- granted below with grant option

-- allow this user to create users
GRANT USER ADMIN TO <SCHEMA>_HTA_USER_ADMIN; --WITH ADMIN OPTION;

-- grant some object privileges which are needed internally by the procedures
GRANT SELECT ON SYS.DUMMY TO <SCHEMA>_HTA_USER_ADMIN WITH GRANT OPTION;
GRANT SELECT ON SYS.USERS TO <SCHEMA>_HTA_USER_ADMIN WITH GRANT OPTION;
GRANT SELECT ON SYS.M_PASSWORD_POLICY TO <SCHEMA>_HTA_USER_ADMIN WITH GRANT OPTION;
GRANT EXECUTE ON SYS.IS_VALID_USER_NAME TO <SCHEMA>_HTA_USER_ADMIN WITH GRANT OPTION;

-- grant some privileges which are granted to the technical users which are created via
CREATE_TECHNICAL_USER
GRANT RESTRICTED_USER_JDBC_ACCESS TO <SCHEMA>_HTA_USER_ADMIN WITH ADMIN OPTION;
GRANT RESTRICTED_USER_ODBC_ACCESS TO <SCHEMA>_HTA_USER_ADMIN WITH ADMIN OPTION;

-- logon with the user to create the procedures
CONNECT <SCHEMA>_HTA_USER_ADMIN PASSWORD abcd1234ABC_D;

-- create the CREATE_PASSWORD procedure for creating a password
CREATE PROCEDURE CREATE_PASSWORD(
  -- create a password
  OUT PASSWORD NVARCHAR(256) -- created password
)
LANGUAGE SQLSCRIPT
SQL SECURITY INVOKER
AS BEGIN
-- length of the password we generate
DECLARE PASSWORD_LENGTH CONSTANT INTEGER = 128;

-- minimum number of upper, lower, digit, and special characters
DECLARE MIN_UPPER_CHARACTERS INTEGER;
DECLARE MIN_LOWER_CHARACTERS INTEGER;
DECLARE MIN_DIGIT_CHARACTERS INTEGER;
DECLARE MIN_SPECIAL_CHARACTERS INTEGER;

-- constants for password generation
DECLARE UPPER_CHARACTERS CONSTANT NVARCHAR(26) = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ';
DECLARE LOWER_CHARACTERS CONSTANT NVARCHAR(26) = 'abcdefghijklmnopqrstuvwxyz';
DECLARE DIGIT_CHARACTERS CONSTANT NVARCHAR(10) = '0123456789';
DECLARE SPECIAL_CHARACTERS CONSTANT NVARCHAR(22) = '^!" $%&/()=?+*#''_-.:;'

-- calculate the minimum number of upper, lower, digit, and special characters
BEGIN
  DECLARE PASSWORD_LAYOUT_LENGTH INTEGER;
  SELECT LENGTH(VALUE),
         OCCURRENCES_REGEXPR('([[:upper:]])' IN VALUE),
         OCCURRENCES_REGEXPR('([[:lower:]])' IN VALUE),
         OCCURRENCES_REGEXPR('([[:digit:]])' IN VALUE)
  INTO PASSWORD_LAYOUT_LENGTH,
      MIN_UPPER_CHARACTERS,
      MIN_LOWER_CHARACTERS,
      MIN_DIGIT_CHARACTERS
  FROM SYS.M_PASSWORD_POLICY WHERE PROPERTY = 'password_layout';
  MIN_SPECIAL_CHARACTERS := PASSWORD_LAYOUT_LENGTH - MIN_UPPER_CHARACTERS -
                           MIN_LOWER_CHARACTERS - MIN_DIGIT_CHARACTERS;
END;

-- start with a fresh password
PASSWORD := '';

-- add the minimum of required characters to the password
BEGIN
  DECLARE I INTEGER;
  FOR I IN 1 .. MIN_UPPER_CHARACTERS DO
    PASSWORD := PASSWORD || SUBSTRING(UPPER_CHARACTERS,
                                       FLOOR(RAND_SECURE() *
                                       LENGTH(UPPER_CHARACTERS)) + 1, 1);
  END FOR;

  FOR I IN 1 .. MIN_LOWER_CHARACTERS DO
    PASSWORD := PASSWORD || SUBSTRING(LOWER_CHARACTERS,
                                       FLOOR(RAND_SECURE() *
                                       LENGTH(LOWER_CHARACTERS)) + 1, 1);
  END FOR;

  FOR I IN 1 .. MIN_DIGIT_CHARACTERS DO
    PASSWORD := PASSWORD || SUBSTRING(DIGIT_CHARACTERS,
                                       FLOOR(RAND_SECURE() *
                                       LENGTH(DIGIT_CHARACTERS)) + 1, 1);
  END FOR;

  FOR I IN 1 .. MIN_SPECIAL_CHARACTERS DO
    PASSWORD := PASSWORD || SUBSTRING(SPECIAL_CHARACTERS,
                                       FLOOR(RAND_SECURE() *
                                       LENGTH(SPECIAL_CHARACTERS)) + 1, 1);
  END FOR;

END;

-- fill up with random characters
BEGIN
  DECLARE ALL_CHARACTERS CONSTANT NVARCHAR(84) = UPPER_CHARACTERS || LOWER_CHARACTERS || DIGIT_CHARACTERS || SPECIAL_CHARACTERS;
  DECLARE I INTEGER;
  FOR I IN LENGTH(PASSWORD) + 1 .. PASSWORD_LENGTH DO
    PASSWORD := PASSWORD || SUBSTRING(ALL_CHARACTERS,
                                       FLOOR(RAND_SECURE() *
                                       LENGTH(ALL_CHARACTERS)) + 1, 1);
  END FOR;
END;
-- create the CHECK_VALID_USER_NAME procedure for checking for a valid user name

CREATE PROCEDURE CHECK_VALID_USER_NAME(
    -- check whether the given name is a valid user name
    IN USER_NAME NVARCHAR(127) -- name of the user to be checked
)
LANGUAGE SQLSCRIPT
SQL SECURITY INVOKER
AS
BEGIN
    DECLARE ERROR_INVALID_USER_NAME CONDITION FOR SQL_ERROR_CODE 10001;
    DECLARE ERROR_CODE INTEGER;
    DECLARE ERROR_MESSAGE NVARCHAR(128);
    IF (UPPER(TRIM(:USER_NAME)) != :USER_NAME) THEN
        SIGNAL ERROR_INVALID_USER_NAME SET MESSAGE_TEXT = 'invalid user name: user name contains leading or trailing whitespace or is not in upper case';
    END IF;
    CALL SYS.IS_VALID_USER_NAME(:USER_NAME, :ERROR_CODE, :ERROR_MESSAGE);
    IF (:ERROR_CODE != 0) THEN
        SIGNAL ERROR_INVALID_USER_NAME SET MESSAGE_TEXT = 'invalid user name: ' || ERROR_MESSAGE;
    END IF;
END;

-- create the CREATE_TECHNICAL_USER procedure

CREATE PROCEDURE CREATE_TECHNICAL_USER(
    -- create a technical user
    IN USER_NAME NVARCHAR(127), -- name of the user to be created
    OUT USER_PASSWORD NVARCHAR(128) -- password of the user to be created
)
LANGUAGE SQLSCRIPT
SQL SECURITY DEFINER
AS
BEGIN
    -- check USER_NAME
    CALL CHECK_VALID_USER_NAME(:USER_NAME);
    -- create a password
    CALL CREATE_PASSWORD(:USER_PASSWORD);
    -- create the technical user
    EXEC('CREATE RESTRICTED USER ' || :USER_NAME || ' PASSWORD "' || ESCAPE_DOUBLE_QUOTES(:USER_PASSWORD) || '"';
    EXEC('ALTER USER ' || :USER_NAME || ' DISABLE PASSWORD LIFETIME');
    -- allow connect
    EXEC('ALTER USER ' || :USER_NAME || ' ENABLE CLIENT CONNECT');
    EXEC('GRANT RESTRICTED_USER_JDBC_ACCESS TO ' || :USER_NAME);
    EXEC('GRANT RESTRICTED_USER_ODBC_ACCESS TO ' || :USER_NAME);
    -- grant some additional roles as required for this technical user
    EXEC('GRANT <ROLE> TO ' || :USER_NAME);
END;
-- grant EXECUTE privileges on CREATE_TECHNICAL_USER to USER
GRANT EXECUTE ON CREATE_TECHNICAL_USER TO <USER>;

-- create the DROP_TECHNICAL_USER procedure
CREATE PROCEDURE DROP_TECHNICAL_USER(
    -- drops a technical user
    IN USER_NAME NVARCHAR(127) -- name of the user to drop
)
LANGUAGE SQLSCRIPT
SQL SECURITY DEFINER
AS BEGIN
    DECLARE ERROR_INVALID_USER CONDITION FOR SQL_ERROR_CODE 10002;

    -- check USER_NAME
    CALL CHECK_VALID_USER_NAME(:USER_NAME);

    -- check that the user was created by the owner of this procedure
    BEGIN
        DECLARE USER_WAS_CREATED_BY_SELF INTEGER;
        SELECT COUNT(*) INTO USER_WAS_CREATED_BY_SELF FROM SYS.USERS WHERE USER_NAME = :USER_NAME AND CREATOR = CURRENT_USER;
        IF (:USER_WAS_CREATED_BY_SELF = 0) THEN
            SIGNAL ERROR_INVALID_USER SET MESSAGE_TEXT = 'invalid user';
        END IF;
    END;

    -- drop the user
    EXEC 'DROP USER ' || :USER_NAME || ' CASCADE';
END;

-- grant EXECUTE privileges on DROP_TECHNICAL_USER to USER
GRANT EXECUTE ON DROP_TECHNICAL_USER TO <USER>;

-- login with a USER ADMIN user
CONNECT SYSTEM PASSWORD <manager>;

-- deactivate the technical owner of the procedures and remove some privileges
ALTER USER <SCHEMA>_HTA_USER_ADMIN DEACTIVATE;
ALTER USER <SCHEMA>_HTA_USER_ADMIN DISABLE CLIENT CONNECT;
--REVOKE RESTRICTED_USER_JDBC_ACCESS FROM <SCHEMA>_HTA_USER_ADMIN; -- still needed
--REVOKE RESTRICTED_USER_ODBC_ACCESS FROM <SCHEMA>_HTA_USER_ADMIN; -- still needed

5.2 Script for Creating an SAP HANA Role

-- How to Use this script!
-- 0. If this is a MDC/MultiDB HANA system, connect to the tenant DB and NOT to the SYSTEMDB
-- 1. Set correct password for user SYSTEM by replacing all occurrences of <manager>!!!
-- 2. Select all statements and execute them in one shot
-- 3. If there were errors reported during GRANTS, they can be ignored because depending on HANA version there are more/less views/table/procedures available.
-- 4. In case you are in doubt, create ticket on BC-CTS-HTA

-- connect as SYSTEM user
CONNECT SYSTEM PASSWORD '<manager>';

CREATE role HTA_UPS_COMMON;

GRANT SELECT ON _SYS_BI.BIMC_ALL_AUTHORIZED_CUBES TO HTA_UPS_COMMON;
GRANT SELECT ON _SYS_BI.BIMC_ALL_CUBES HDI TO HTA_UPS_COMMON;
GRANT SELECT ON _SYS_BI.BIMC_ALL_SOURCES TO HTA_UPS_COMMON;
GRANT SELECT ON _SYS_BI.BIMC_ALL_VARIABLES_VIEW TO HTA_UPS_COMMON;
GRANT SELECT ON _SYS_BI.BIMC_ATTRIBUTE_RELATIONS TO HTA_UPS_COMMON;
GRANT SELECT ON _SYS_BI.BIMC_CUBES TO HTA_UPS_COMMON;
GRANT SELECT ON _SYS_BI.BIMC_DIMENSION_VIEW TO HTA_UPS_COMMON;
GRANT SELECT ON _SYS_BI.BIMC_DIMENSION_VIEW HDI TO HTA_UPS_COMMON;
GRANT SELECT ON _SYS_BI.BIMC_DIMENSIONS TO HTA_UPS_COMMON;
GRANT SELECT ON _SYS_BI.BIMC_DIMENSIONS HDI TO HTA_UPS_COMMON;
GRANT SELECT ON _SYS_BI.BIMC_HIERARCHIES TO HTA_UPS_COMMON;
GRANT SELECT ON _SYS_BI.BIMC_HIERARCHIES HDI TO HTA_UPS_COMMON;
GRANT SELECT ON _SYS_BI.BIMC_MEASURES TO HTA_UPS_COMMON;
GRANT SELECT ON _SYS_BI.BIMC_MEASURES HDI TO HTA_UPS_COMMON;
GRANT SELECT ON _SYS_BI.BIMC_PROPERTIES_VIEW TO HTA_UPS_COMMON;
GRANT SELECT ON _SYS_BI.BIMC_VARIABLE_MAPPING TO HTA_UPS_COMMON;
GRANT SELECT ON _SYS_BI.BIMC_VARIABLE_RANGE_DEFAULTS TO HTA_UPS_COMMON;
GRANT SELECT ON _SYS_BI.BIMC_VARIABLE_VALUE TO HTA_UPS_COMMON;
GRANT SELECT ON _SYS_BI.BIMC_VARIABLE_VIEW TO HTA_UPS_COMMON;
GRANT SELECT ON _SYS_BI.BIMC_VARIABLE_VIEW HDI TO HTA_UPS_COMMON;
GRANT SELECT ON _SYS_SQL_ANALYZER.OPERATOR_STATISTICS TO HTA_UPS_COMMON;
GRANT SELECT ON _SYS_SQL_ANALYZER.STATEMENT_STATISTICS TO HTA_UPS_COMMON;
GRANT SELECT ON _SYS_TASK.BEST_RECORD_GROUP_MASTER_STATISTICS TO HTA_UPS_COMMON;
GRANT SELECT ON _SYS_TASK.BEST_RECORD_RESULTS TO HTA_UPS_COMMON;
GRANT SELECT ON _SYS_TASK.BEST_RECORD_STRATEGIES TO HTA_UPS_COMMON;
GRANT SELECT ON _SYS_TASK.CLEANUP_TASK_DATA TO HTA_UPS_COMMON;
GRANT SELECT ON _SYS_TASK.CREATE_TASK_MESSAGES TO HTA_UPS_COMMON;
GRANT SELECT ON _SYS_TASK.DQ_REFERENCEDATA_EXPIRY_RESULT TO HTA_UPS_COMMON;
GRANT SELECT ON _SYS_TASK.GEOCODE_INFO_CODES TO HTA_UPS_COMMON;
GRANT SELECT ON _SYS_TASK.GEOCODE_INFO_C TO HTA_UPS_COMMON;
GRANT SELECT ON _SYS_TASK.GEOCODE_RESULT TO HTA_UPS_COMMON;
GRANT SELECT ON _SYS_TASK.MATCH_GROUP_INFO TO HTA_UPS_COMMON;
GRANT SELECT ON _SYS_TASK.MATCH_RECORD_INFO TO HTA_UPS_COMMON;
GRANT SELECT ON _SYS_TASK.MATCH_SOURCE_STATISTICS TO HTA_UPS_COMMON;
GRANT SELECT ON _SYS_TASK.MATCH_STATISTICS TO HTA_UPS_COMMON;
GRANT SELECT ON _SYS_TASK.MATCH_TRACING TO HTA_UPS_COMMON;
GRANT EXECUTE ON _SYS_TASK.PROFILE_FREQUENCY_DISTRIBUTION TO HTA_UPS_COMMON;
GRANT SELECT ON _SYS_TASK.PROFILE_FREQUENCY_DISTRIBUTION_COLUMNS TO HTA_UPS_COMMON;
GRANT SELECT ON _SYS_TASK.PROFILE_FREQUENCY_DISTRIBUTION_RESULT TO HTA_UPS_COMMON;
GRANT EXECUTE ON _SYS_TASK.PROFILE_METADATA TO HTA_UPS_COMMON;
GRANT SELECT ON _SYS_TASK.PROFILE_METADATA_COLUMNS TO HTA_UPS_COMMON;
GRANT EXECUTE ON _SYS_TASK.PROFILE_METADATA_RESULT TO HTA_UPS_COMMON;
GRANT SELECT ON _SYS_TASK.PROFILE_SEMANTIC_TO HTA_UPS_COMMON;
GRANT SELECT ON _SYS_TASK.PROFILE_SEMANTIC_COLUMNS TO HTA_UPS_COMMON;
GRANT SELECT ON _SYS_TASK.PROFILE_SEMANTIC_RESULT TO HTA_UPS_COMMON;
GRANT SELECT ON _SYS_TASK.START_TASK_MESSAGES TO HTA_UPS_COMMON;
GRANT SELECT ON _SYS_TASK.TASK_CLIENT_MAPPING TO HTA_UPS_COMMON;
GRANT SELECT ON _SYS_TASK.TASK_COLUMN_DEFINITIONS TO HTA_UPS_COMMON;
GRANT SELECT ON SYS_TASK.TASK_EXECUTIONS TO HTA_UPS_COMMON;
GRANT SELECT ON SYS_TASK.TASK_LOCALIZATION TO HTA_UPS/Common;
GRANT SELECT ON SYS_TASK.TASK_OPERATIONS TO HTA_UPS_COMMON;
GRANT SELECT ON SYS_TASK.TASK_OPERATIONS_EXECUTIONS TO HTA_UPS_COMMON;
GRANT SELECT ON SYS_TASK.TASK_TABLE_DEFINITIONS TO HTA_UPS_COMMON;
GRANT SELECT ON SYS_TASK.TASK_TABLE_RELATIONSHIPS TO HTA_UPS_COMMON;
GRANT SELECT ON SYS.ACCESSIBLE_VIEWS TO HTA_UPS_COMMON;
GRANT EXECUTE ON SYS_ADVANCED_PLANVIZ_ACTION TO HTA_UPS_COMMON;
GRANT SELECT ON SYS_ASSOCIATIONS TO HTA_UPS_COMMON;
GRANT SELECT ON SYS_CDS_ANNOTATION_ASSIGNMENTS TO HTA_UPS_COMMON;
GRANT SELECT ON SYS_CDS_ANNOTATION_VALUES TO HTA_UPS_COMMON;
GRANT EXECUTE ON SYS_CDS_ARTIFACT_DEFINITION TO HTA_UPS_COMMON;
GRANT SELECT ON SYS_CDS_ARTIFACT_NAMES TO HTA_UPS_COMMON;
GRANT SELECT ON SYS_CDS_ASSOCIATIONS TO HTA_UPS_COMMON;
GRANT SELECT ON SYS_CDS_ENTITIES TO HTA_UPS_COMMON;
GRANT EXECUTE ON SYS_CDS_SCHEMA_LAST_MODIFIED_TIME TO HTA_UPS_COMMON;
GRANT SELECT ON SYS_CDS_VIEWS TO HTA_UPS_COMMON;
GRANT SELECT ON SYS_CONSTRAINTS TO HTA_UPS_COMMON;
GRANT EXECUTE ON SYS_CREATE_INTERMEDIATE_CALCULATION_VIEW_DEV TO HTA_UPS_COMMON;
GRANT SELECT ON SYS_DATA_TYPES TO HTA_UPS_COMMON;
GRANT EXECUTE ON SYS_DEBUG TO HTA_UPS_COMMON;
GRANT EXECUTE ON SYS_DSO_ACTIVATE_CHANGES TO HTA_UPS_COMMON;
GRANT EXECUTE ON SYS_DSO_Rollback_CHANGES TO HTA_UPS_COMMON;
GRANT SELECT ON SYS_DUMMY TO HTA_UPS_COMMON;
GRANT SELECT ON SYS_EFFECTIVE_PRIVILEGES TO HTA_UPS_COMMON;
GRANT SELECT ON SYS_EFFECTIVE_ROLES TO HTA_UPS_COMMON;
GRANT SELECT ON SYS_EFFECTIVE_STRUCTURED_PRIVILEGES TO HTA_UPS_COMMON;
GRANT SELECT ON SYS_EXPLAIN_PLAN_TABLE TO HTA_UPS_COMMON;
GRANT SELECT ON SYS_Flexible_TABLES TO HTA_UPS_COMMON;
GRANT SELECT ON SYS_FullTEXT_INDEXES TO HTA_UPS_COMMON;
GRANT SELECT ON SYS_FUNCTION_PARAMETER_COLUMNS TO HTA_UPS_COMMON;
GRANT SELECT ON SYS_FUNCTION_PARAMETERS TO HTA_UPS_COMMON;
GRANT SELECT ON SYS_FUNCTIONS TO HTA_UPS_COMMON;
GRANT SELECT ON SYS_GEOCODE_INDEXES TO HTA_UPS_COMMON;
GRANT EXECUTE ON SYS_GET_JOIN_INFO TO HTA_UPS_COMMON;
GRANT EXECUTE ON SYS_GET_OBJECT_DEFINITION TO HTA_UPS_COMMON;
GRANT EXECUTE ON SYS_GET_REMOTE_SOURCE_COLUMNS TO HTA_UPS_COMMON;
GRANT EXECUTE ON SYS_GET_REMOTE_SOURCE_CS_VIEW_PARAMETERS TO HTA_UPS_COMMON;
GRANT EXECUTE ON SYS_GET_REMOTE_SOURCE_FUNCTION_DEFINITION TO HTA_UPS_COMMON;
GRANT EXECUTE ON SYS_GET_REMOTE_SOURCE_OBJECT_PARAMETERS TO HTA_UPS_COMMON;
GRANT EXECUTE ON SYS_GET_REMOTE_SOURCE_OBJECT_TREE TO HTA_UPS_COMMON;
GRANT EXECUTE ON SYS_GET_REMOTE_SOURCE_OBJECTS TO HTA_UPS_COMMON;
GRANT EXECUTE ON SYS_GET_REMOTE_SOURCE_OBJECTS_LIST TO HTA_UPS_COMMON;
GRANT EXECUTE ON SYS_GET_REMOTE_SOURCE_TABLE_DEFINITIONS TO HTA_UPS_COMMON;
GRANT EXECUTE ON SYS_GET_REMOTE_SOURCE_TABLE_ESS_DEFINITIONS TO HTA_UPS_COMMON;
GRANT SELECT ON SYS_GRANTED_PRIVILEGES TO HTA_UPS_COMMON;
GRANT SELECT ON SYS_GRANTED_ROLES TO HTA_UPS_COMMON;
GRANT SELECT ON SYS_INDEX_WORKSPACES TO HTA_UPS_COMMON;
GRANT SELECT ON SYS_INDEX_COLUMNS TO HTA_UPS_COMMON;
GRANT SELECT ON SYS_INDEXES TO HTA_UPS_COMMON;
GRANT SELECT ON SYS_Libraries TO HTA_UPS_COMMON;
GRANT SELECT ON SYS_M_CONNECTIONS TO HTA_UPS_COMMON;
GRANT SELECT ON SYS_M_CS_PARTITIONS TO HTA_UPS_COMMON;
GRANT SELECT ON SYS_M_CS_TABLES TO HTA_UPS_COMMON;
GRANT SELECT ON SYS_M_DATABASE TO HTA_UPS_COMMON;
GRANT SELECT ON SYS_M_DEBUG_SESSIONS TO HTA_UPS_COMMON;
GRANT SELECT ON SYS_M_ES_TABLES TO HTA_UPS_COMMON;
GRANT SELECT ON SYS_M_FEATURES TO HTA_UPS_COMMON;
GRANT SELECT ON SYS.M_FULLTEXT_QUEUES TO HTA_UPS_COMMON;
GRANT SELECT ON SYS.M_HOST_INFORMATION TO HTA_UPS_COMMON;
GRANT SELECT ON SYS.M_MONITOR_COLUMNS TO HTA_UPS_COMMON;
GRANT SELECT ON SYS.M_MONITORS TO HTA_UPS_COMMON;
GRANT SELECT ON SYS.M_RS_TABLES TO HTA_UPS_COMMON;
GRANT SELECT ON SYS.M_SESSION_CONTEXT TO HTA_UPS_COMMON;
GRANT SELECT ON SYS.M_SYSTEM_OVERVIEW TO HTA_UPS_COMMON;
GRANT SELECT ON SYS.M_TABLES TO HTA_UPS_COMMON;
GRANT SELECT ON SYS.M_TEMPORARY_TABLES TO HTA_UPS_COMMON;
GRANT SELECT ON SYS.M_TEXT_ANALYSIS_LANGUAGES TO HTA_UPS_COMMON;
GRANT SELECT ON SYS.OBJECT_DEPENDENCIES TO HTA_UPS_COMMON;
GRANT SELECT ON SYS.OBJECT_PRIVILEGES TO HTA_UPS_COMMON;
GRANT SELECT ON SYS.OBJECTS TO HTA_UPS_COMMON;
GRANT SELECT ON SYS.OPTIONS TO HTA_UPS_COMMON;
GRANT SELECT ON SYS_PARTITIONED_TABLES TO HTA_UPS_COMMON;
GRANT EXECUTE ON SYS.PLANVIZ_ACTION TO HTA_UPS_COMMON;
GRANT SELECT ON SYS.PRIVILEGES TO HTA_UPS_COMMON;
GRANT SELECT ON SYS.PROCEDURE_PARAMETER_COLUMNS TO HTA_UPS_COMMON;
GRANT SELECT ON SYS.PROCEDURE_PARAMETERS TO HTA_UPS_COMMON;
GRANT SELECT ON SYS.PROCEDURES TO HTA_UPS_COMMON;
GRANT SELECT ON SYS.PROJECTION_VIEW_COLUMN_SOURCES TO HTA_UPS_COMMON;
GRANT SELECT ON SYS.REFERENTIAL_CONSTRAINTS TO HTA_UPS_COMMON;
GRANT SELECT ON SYS.REMOTE_SOURCE_OBJECT_COLUMN_CONSTRAINTS TO HTA_UPS_COMMON;
GRANT SELECT ON SYS.REMOTE_SOURCE_OBJECT_COLUMN_DESCRIPTIONS TO HTA_UPS_COMMON;
GRANT SELECT ON SYS.REMOTE_SOURCE_OBJECT_COLUMNS TO HTA_UPS_COMMON;
GRANT SELECT ON SYS.REMOTE_SOURCE_OBJECTS TO HTA_UPS_COMMON;
GRANT SELECT ON SYS.REMOTE_SOURCES TO HTA_UPS_COMMON;
GRANT SELECT ON SYS.REMOTE_SUBSCRIPTION_EXCEPTIONS TO HTA_UPS_COMMON;
GRANT SELECT ON SYS.REMOTE_SUBSCRIPTIONS TO HTA_UPS_COMMON;
GRANT SELECT ON SYS.RESERVED_KEYWORDS TO HTA_UPS_COMMON;
GRANT SELECT ON SYS.RESULT_CACHE TO HTA_UPS_COMMON;
GRANT SELECT ON SYS.RESULT_CACHE_COLUMNS TO HTA_UPS_COMMON;
GRANT SELECT ON SYS.SCHEMAS TO HTA_UPS_COMMON;
GRANT SELECT ON SYS.sequences TO HTA_UPS_COMMON;
GRANT EXECUTE ON SYS.SERIES_GENERATE_BIGINT TO HTA_UPS_COMMON;
GRANT EXECUTE ON SYS.SERIES_GENERATE_DATE TO HTA_UPS_COMMON;
GRANT EXECUTE ON SYS.SERIES_GENERATE_DECIMAL TO HTA_UPS_COMMON;
GRANT EXECUTE ON SYS.SERIES_GENERATE_INTEGER TO HTA_UPS_COMMON;
GRANT EXECUTE ON SYS.SERIES_GENERATE_SECONDDATE TO HTA_UPS_COMMON;
GRANT EXECUTE ON SYS.SERIES_GENERATE_SMALLDECIMAL TO HTA_UPS_COMMON;
GRANT EXECUTE ON SYS.SERIES_GENERATE_SMALLINT TO HTA_UPS_COMMON;
GRANT EXECUTE ON SYS.SERIES_GENERATE_TIMESTAMP TO HTA_UPS_COMMON;
GRANT EXECUTE ON SYS.SERIES_GENERATE_TINYINT TO HTA_UPS_COMMON;
GRANT SELECT ON SYS.SERIES_KEY_COLUMNS TO HTA_UPS_COMMON;
GRANT SELECT ON SYS.SERIES_TABLES TO HTA_UPS_COMMON;
GRANT SELECT ON SYS.ST_GEOMETRY_COLUMNS TO HTA_UPS_COMMON;
GRANT SELECT ON SYS.ST_SPATIAL_REFERENCE_SYSTEMS TO HTA_UPS_COMMON;
GRANT SELECT ON SYS.S消除_units_of_measure TO HTA_UPS_COMMON;
GRANT SELECT ON SYS.ST_STRUCTURED_PRIVILEGES TO HTA_UPS_COMMON;
GRANT EXECUTE ON SYS.TA_ANALYZE TO HTA_UPS_COMMON;
GRANT SELECT ON SYS.TA_COLUMNS TO HTA_UPS_COMMON;
GRANT SELECT ON SYS.TA_COLUMNS_ODBC TO HTA_UPS_COMMON;
GRANT SELECT ON SYS.TA_GROUPS TO HTA_UPS_COMMON;
GRANT SELECT ON SYS.TA_PARTITIONS TO HTA_UPS_COMMON;
GRANT SELECT ON SYS.TAL_PLACE TO HTA_UPS_COMMON;
GRANT SELECT ON SYS.TA_REPLICAS TO HTA_UPS_COMMON;
GRANT SELECT ON SYS.TABLES TO HTA_UPS_COMMON;
GRANT SELECT ON SYS.TASK_PARAMETERS TO HTA_UPS_COMMON;
GRANT SELECT ON SYS.TASKS TO HTA_UPS_COMMON;
GRANT EXECUTE ON SYS.TEXT_CONFIGURATION_CLEAR TO HTA_UPS_COMMON;
GRANT EXECUTE ON SYS.TEXT_CONFIGURATION_CREATE TO HTA_UPS_COMMON;
GRANT EXECUTE ON SYS.TEXT_CONFIGURATION_DROP TO HTA_UPS_COMMON;
GRANT SELECT ON SYS.TEXT_CONFIGURATIONS TO HTA_UPS_COMMON;
GRANT SELECT ON SYS.TIMEZONES TO HTA_UPS_COMMON;
GRANT SELECT ON SYS.TRIGGERS TO HTA_UPS_COMMON;
GRANT SELECT ON SYS.USER_PARAMETERS TO HTA_UPS_COMMON;
GRANT EXECUTE ON SYS.VALIDATE_EXPRESSION TO HTA_UPS_COMMON;
GRANT SELECT ON SYS.VIEW_COLUMNS TO HTA_UPS_COMMON;
GRANT SELECT ON SYS.VIEWS TO HTA_UPS_COMMON;
GRANT SELECT ON SYS.VIRTUAL_COLUMN_PROPERTIES TO HTA_UPS_COMMON;
GRANT SELECT ON SYS.VIRTUAL_COLUMNS TO HTA_UPS_COMMON;
GRANT SELECT ON SYS.VIRTUAL_FUNCTION_PACKAGES TO HTA_UPS_COMMON;
GRANT SELECT ON SYS.VIRTUAL_FUNCTIONS TO HTA_UPS_COMMON;
GRANT SELECT ON SYS.VIRTUAL_PROCEDURES TO HTA_UPS_COMMON;
GRANT SELECT ON SYS.VIRTUAL_TABLE_PARAMETERS TO HTA_UPS_COMMON;
GRANT SELECT ON SYS.VIRTUAL_TABLE_PROPERTIES TO HTA_UPS_COMMON;
GRANT SELECT ON SYS.VIRTUAL_TABLES TO HTA_UPS_COMMON;

5.3 Configuring Secure Connection from ABAP to XSA

Your ABAP server needs to connect to the XSA server with secure http connections and therefore needs to accept the SSL certificate of the XSA server. To enable ABAP to trust XSA certificates you need to either import the XSA certificate or one of his parents in the certification path. If this is already for your system, you can skip this chapter

1. Log on to your XSA server via the command line tool xs. Use the database user that was used when installing XSA.
2. Execute the command xs login -s SAP
   Log on with the user XSA_ADMIN and enter the password

```
Adabas: [Username] net/sap/HDS/NU0925:0310280> xs login -s SAP
API URL: https://[Username]:30280
USERNAME: XSA_ADMIN
PASSWORD: 
Authenticating...
Org: AT5ORG
SPACE: SAP
API endpoint: https://[Username]:30280 (API version: 1)
User: XSA_ADMIN
Org: AT5ORG
Space: SAP
```
3. Execute the command `xs -v`. Search for the URL for `authorizationEndpoint` and copy it.

4. Extend the URL with `/oauth/token`. The final URL looks like this: `https://<server>:<port>/uaa-security/oauth/token`

In the reminder of this chapter, we use google chrome. If you use another browser, some features (like finding the certificate) work differently.

Scroll down to the end of this chapter to learn how to get the certificate when using another browser.

5. Copy the URL to a browser window. Click `Cancel` on the authentication pop-up.
6. Press F12. Click on Security, then on View certificate and then in the pop-up window on Certification Path. Make sure that the certificate is ok.

7. Download one of the three certificates e.g. the certificate of the XSA server – click on Details and choose Copy to File…
8. Go through the wizard. Click Next.

9. Leave the settings as they are (or choose the format you need) and click Next.
10. Browse for the location where you would like to store the file and name it <file>.cer.

11. Check that everything is correct and click Finish.
12. Click ok

13. Log on to your ABAP system and open transaction STRUST

14. If the section SSL client SSL Client (Anonymous) is marked with a red cross, continue with the following steps (15 to 17). If SSL in general is already configured (and you can see a folder symbol in front of SSL client SSL Client (Anonymous)) continue with step 18.
15. Switch to change mode (Choose Display <-> Change). Right click on SSL client SSL Client (Anonymous). Choose Create.

16. Leave everything as it is in the following pop-up or change the settings according to your needs. Check SAP Note 510007 for details. Click on Continue.

17. You should see a success message at the bottom of the screen. The section SSL client SSL Client (Anonymous) should now show a folder.
18. Now the folder **SSL client SSL Client (Anonymous)** should exist but there is still no certificate shown in the **Certificate List** area.

19. Choose **Certificate → Import**
20. Browse for the location where you stored the certificate before and click on *Continue*.

21. Allow the system to access the file.
22. Choose Add to Certificate List.
23. Now, the certificate is shown in the Certificate List. Click on Save.

24. Make sure the node SSL client SSL (Standard) is shown as a folder and has subnodes. If not, switch to change mode (Choose Display -> Change). Right click on SSL client SSL Client (Standard). Choose Create and Confirm the popup.

**NOTE**

To download the certificate in different browsers, do the following:

- **Internet Explorer** (Run as administrator):
  In the same tab, where you opened the link: Start the navigation pathSecurity report
Click on **Certificate**
Choose the tab **Details**
Click on **Copy to File…**
Choose **Next**
On the next screen, leave the settings as they are or choose the format that you need
Choose **Next**
Browser for the folder where you would like to store the certificate. Enter <your file name>
Choose **Next**
Choose **Finish**
Confirm the pop-up that the file was saved successfully
Click **OK**

- **Firefox**
  In the same tab, where you opened the link, start the navigation path
  Show site information
  Click on the **Arrow**
  Click on **More information**
  Click on **Security**
  Click on **View Certificate**
  Click on **Details**
  Click on **Export…**
  Click on **Save**

- **Safari**
  In the same tab, where you opened the link, choose the padlock symbol in the URL bar
  Choose **Show Certificate**.
  Drag the certificate symbol to your desktop.

### 5.4 Using the Report SCTS_HTA_CONFIGURATION

You can define several HDI parameters by using the report SCTS_HTA_CONFIGURATION. In a standard configuration, you only need this report to define the URL for the Web IDE (see chapter [Maintaining the Web IDE URL on ABAP](#)).

If you did use other names for the destinations (chapter [Creating Destinations in ABAP](#)), the XSA space (chapter [Creating XSA Space ABAP_SAP<ABAP_SID>](#)) or when creating the database procedures (chapter [Creating Database Procedures Required by AS ABAP](#)) than the ones suggested in this guide, you have to enter these names in the report as well.

Make sure that you enter the parameters in the correct case.

#### Report SCTS_HTA_CONFIGURATION

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>WebIDE URL</td>
<td><a href="https://www.example.com:52075">https://www.example.com</a></td>
</tr>
<tr>
<td>XSA Space</td>
<td></td>
</tr>
<tr>
<td>CreateUserProcedure</td>
<td></td>
</tr>
<tr>
<td>DB Host</td>
<td></td>
</tr>
<tr>
<td>DB Port</td>
<td></td>
</tr>
<tr>
<td>DB Name</td>
<td></td>
</tr>
<tr>
<td>DBX Destination</td>
<td></td>
</tr>
<tr>
<td>Administration Destination</td>
<td>HUGO</td>
</tr>
<tr>
<td>Http Trace Switch</td>
<td></td>
</tr>
<tr>
<td>Internal: No OAuth flow</td>
<td></td>
</tr>
</tbody>
</table>

### 5.5 Abbreviations

<p>| SUM  | Software Update Manager |</p>
<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SWPM</td>
<td>Software Provisioning Manager</td>
</tr>
<tr>
<td>XSA</td>
<td>SAP HANA XS Advanced</td>
</tr>
<tr>
<td>IDE / Web IDE</td>
<td>SAP Web IDE for SAP HANA</td>
</tr>
<tr>
<td>AMHC</td>
<td>ABAP Managed HDI Container</td>
</tr>
<tr>
<td>HDI</td>
<td>SAP HANA Deployment Infrastructure</td>
</tr>
<tr>
<td>HTA</td>
<td>HANA Transport for ABAP</td>
</tr>
<tr>
<td>ADT</td>
<td>ABAP Development Tools</td>
</tr>
<tr>
<td>AIE</td>
<td>ABAP in Eclipse</td>
</tr>
<tr>
<td>CDS</td>
<td>Core Data Services</td>
</tr>
<tr>
<td>AMDP</td>
<td>ABAP managed database procedures</td>
</tr>
</tbody>
</table>