# S/4HANA
# Extending with External Data

## SELF-GUIDED DEMONSTRATION

**Version 1.1**
September 12, 2017

## Demo purpose

This script shows how to extend S/4HANA by adding data to a CDS virtual data model from an external source. The external data source in this demo will be Hadoop.

## Using the script

The demo is presented in a four-column format. This format is designed to accommodate your time restrictions and interests. The script is meant to be followed from beginning to end.

| Step | What to do | What you should see | Notes |
|---|---|---|---|
| Step number | Step by step instructions | Screen-shots of what you'll see | Details about what you are doing/seeing |

**The workflow to add external data to S/4HANA is normally:**

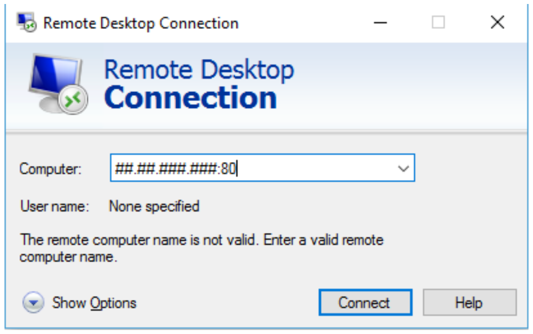
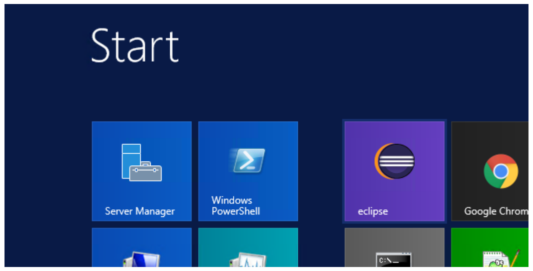
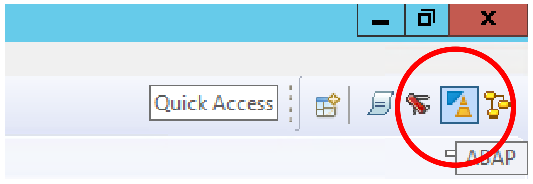
1) Add a Smart Data Access connection in HANA
2) Add a Virtual Table, from the Remote Data Source, in HANA
3) Create a Calculation View off of the HANA Virtual Table
4) Add an External View to the ABAP Data Dictionary (ABAP DDIC)
5) Create a CDS view off of the External View
6) Extend an existing CDS consumption view.

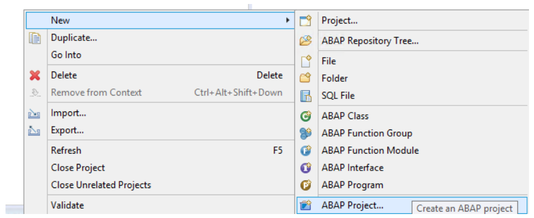
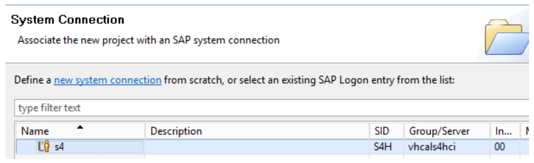
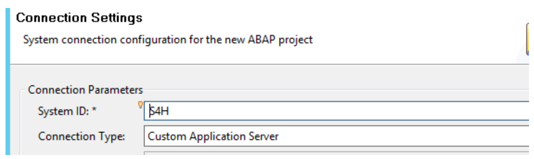
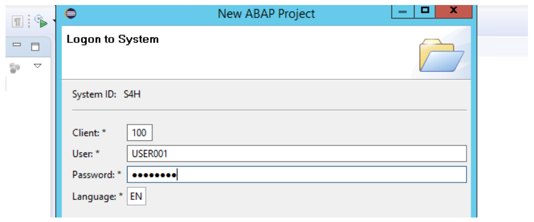
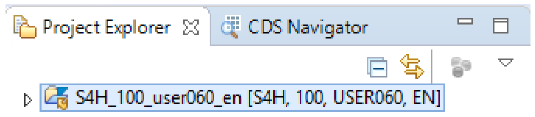
In this particular demo there have been some steps above that are already done. We've already done steps 1 to 3. We have also created a couple of CDS views, one of which is extended by you in step 6.

This script was written for use on S/4HANA 1610 FPS00 or FPS01 On-Premise Edition.

## Preparation

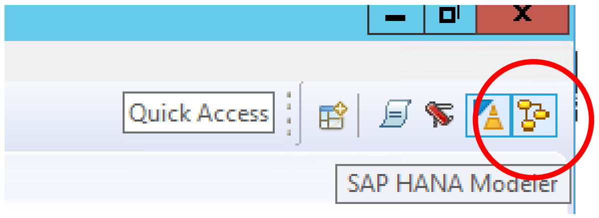
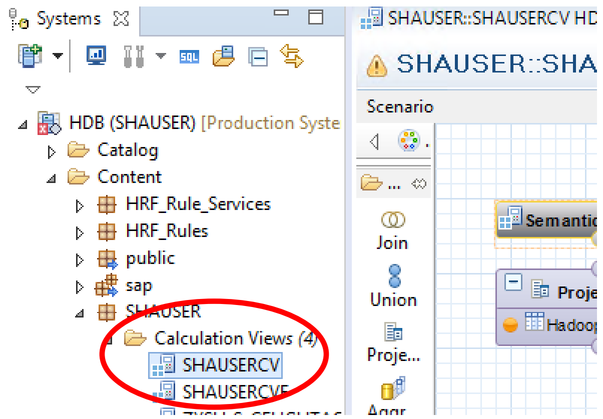
**You need to obtain a Windows System & User ID, plus an S4 User ID from the HANA Academy to follow this workflow.**

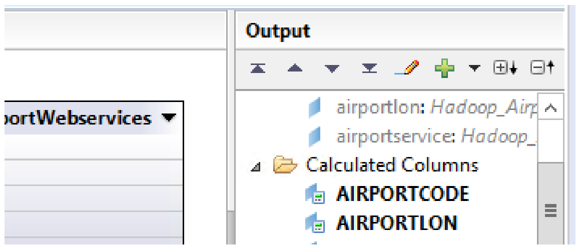
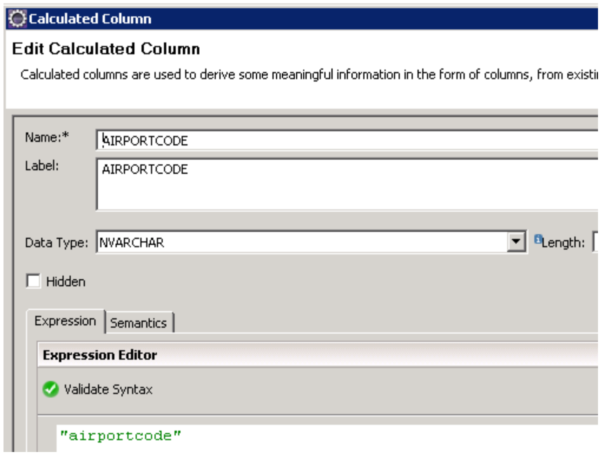
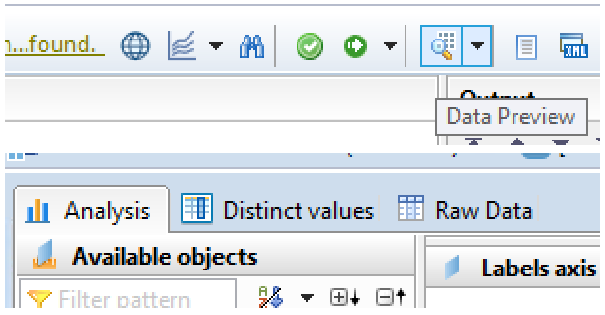
| Step | What to do | What you should see | Notes |
|---|---|---|---|
| P1 | Obtain a User ID from the HANA Academy at the welcome desk. | | You need to be assigned a Windows machine & user as well as an S/4HANA user. |
| P2 | Remote Desktop to your Windows machine using the user password combo you were assigned. |  Remote Desktop Connection<br>Computer: ##.##.###.###:80<br>User name: None specified<br>The remote computer name is not valid. Enter a valid remote computer name.<br> Remote Desktop Connection<br>Show Options<br>Connect<br>Help | Don't forget to put :80 on the end of your IP address for your RDP connection. |
| P3 | On the remote Windows machine click on the Windows icon in the lower left (to open Start) and then click on Eclipse. |   Start<br>Server Manager<br>Windows PowerShell<br>eclipse<br>Google Chrom | Eclipse is required for this exercise as you will be using the ABAP perspective for S/4 CDS views.<br><br>For those who have not worked with CDS views before, you will not actually be programming in ABAP but using a combination of SQL and CDS notations. |
| P4 | If Eclipse does not open by default to the ABAP Perspective, click on the ABAP icon in the top right of Eclipse. |  | |

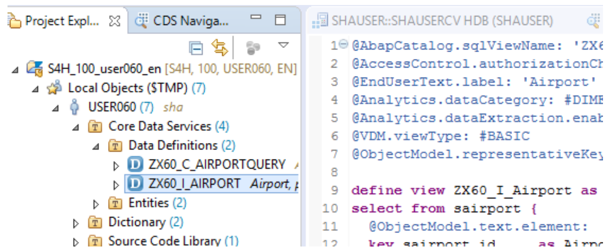
| | | | |
|---|---|---|---|
| P5 | In the Project Explorer on the left right click on the white space and choose New > ABAP Project. |   New<br>Duplicate...<br>Go Into<br>Delete Delete<br>Remove from Context Ctrl+Alt+Shift+Down<br>Import...<br>Export...<br>Refresh F5<br>Close Project<br>Close Unrelated Projects<br>Validate<br>Project...<br>ABAP Repository Tree...<br>File<br>Folder<br>SQL File<br>ABAP Class<br>ABAP Function Group<br>ABAP Function Module<br>ABAP Interface<br>ABAP Program<br>ABAP Project...<br>Create an ABAP project | We are creating a connection to an S4 system. The Windows machine already has a connection to this system created via SAP GUI. |
| P6 | In the System Connection dialogue click on the S4 entry and then click Next.<br><br>In the Connection Settings dialogue click Next to accept the default settings. |  System Connection<br>Associate the new project with an SAP system connection<br>Define a new system connection from scratch, or select an existing SAP Logon entry from the list:<br>type filter text<br>Name: s4 · Description: · SID: S4H · Group/Server: vhcals4hci · In...: 00<br><br> Connection Settings<br>System connection configuration for the new ABAP project<br>Connection Parameters<br>System ID: * S4H<br>Connection Type: Custom Application Server | The next dialogues show the connection details to our existing S4 system. |
| P7 | In the Logon to System dialogue replace the User with the S4 User ID that you were assigned by the HANA Academy.<br><br>Add your user password and click Finish. |  New ABAP Project<br>Logon to System<br>System ID: S4H<br>Client: * 100<br>User: * USER001<br>Password: * ••••••••<br>Language: * EN | If you didn't get an S4 User from the HANA Academy, please do get one now. |
| P8 | You should now see a project for your user in the ABAP Project Explorer. |  S4H_100_user060_en [S4H, 100, USER060, EN] | You are now ready to follow the regular Step by Step demo flow below. |

## *Step-by-step live demo script*

**It is imperative that you have completed all of the steps in the previous Preparation section.**

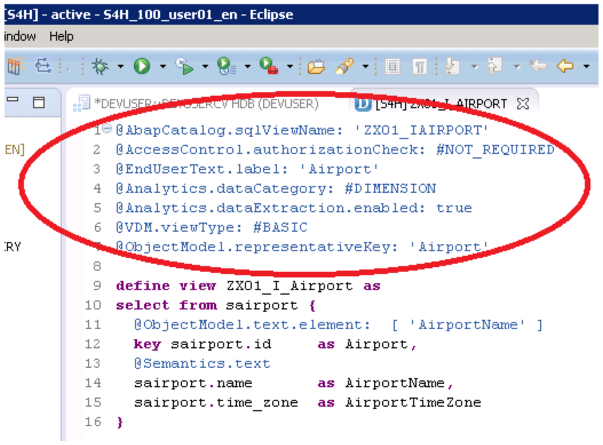
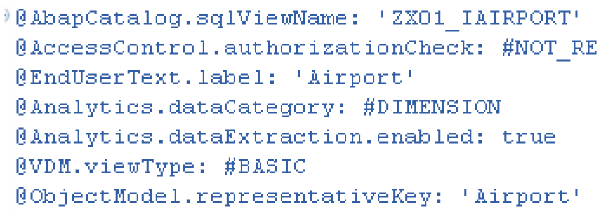
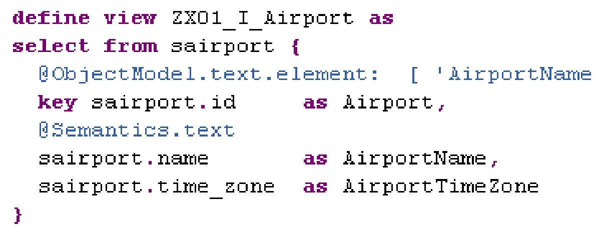
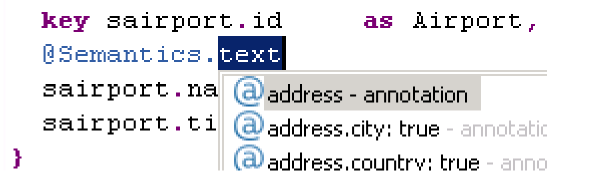
| | What to do | What you should see | Notes |
|---|---|---|---|
| 1 | In this step we will review the existing HANA Calculation View.<br><br>While still in Eclipse open the SAP HANA Modeler perspective.<br><br>In the Systems panel, open the HDB SHAUSER connection and then open the Content folder.<br><br>Navigate to:<br>> SHAUSER package<br>> Calc Views<br>> SHAUSERCV |  <br> Systems<br>HDB (SHAUSER) [Production Syste<br>Catalog<br>Content<br>HRF_Rule_Services<br>HRF_Rules<br>public<br>sap<br>SHAUSER<br>Calculation Views (4)<br>SHAUSERCV<br>SHAUSERCVF<br>SHAUSER::SHAUSERCV HD<br>SHAUSER::SHA<br>Scenario<br>Join<br>Union<br>Proje...<br>Aggr<br>Semantic<br>Proje<br>Hadoo | There is already a HANA Calculation View built on the Virtual Table created with the Smart Data Access connection to the external Hadoop data source.<br><br>If you really want to do the entire workflow from start to finish then you'll be glad to know that we've got videos on the complete process in the HANA Academy. |

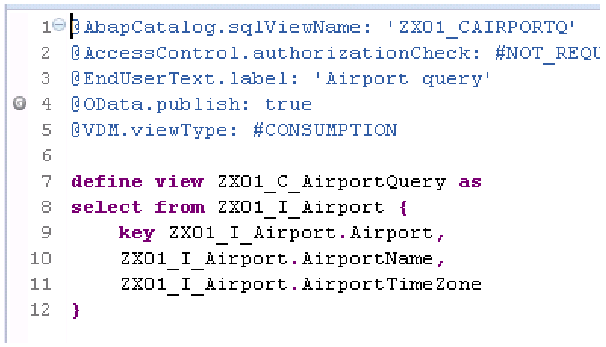
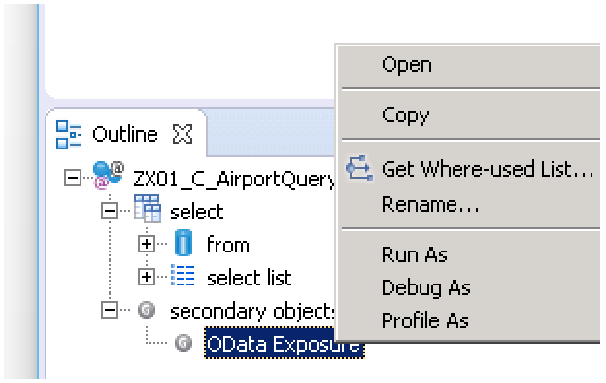
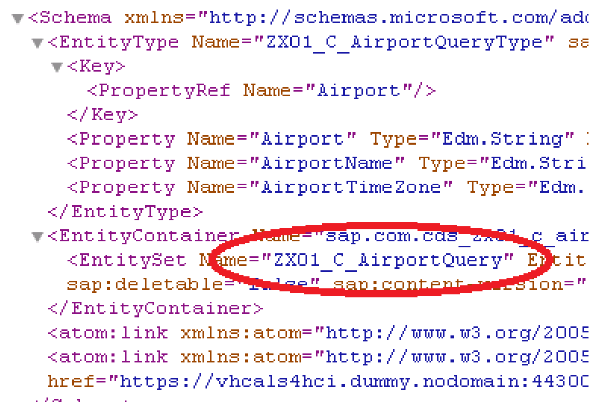
| | What to do | What you should see | Notes |
|---|---|---|---|
| | Click on Projection and note that there is a formula (Calculated Column) for each field that we want in the view output.<br><br>Open up the airport code formula. |  airportlon: *Hadoop_Airp*<br>airportservice: *Hadoop_*<br>Calculated Columns<br>AIRPORTCODE<br>AIRPORTLON<br> Calculated Column<br>Expression<br>Semantics<br>Expression Editor<br>Validate Syntax Edit Calculated Column<br>Calculated columns are used to derive some meaningful information in the form of columns, from existi<br>Name:* AIRPORTCODE<br>Label: AIRPORTCODE<br>Data Type: NVARCHAR Length:<br>☐ Hidden<br>"airportcode" | You may need to use a formula for your output due to issues / errors in some of the next steps. Any field that is replaced by a formula should have the field Hidden so that it does not cause errors later on. Note that these formulae contain only the table fields and no other formula logic.<br><br>One issue is that ABAP External views need upper case field names and in our case the field names are all lower case. To get around this we use upper case formula names.<br><br>We also need to ensure that HANA field types can be converted to ABAP types. Currently there is no conversion from VARCHAR HANA field types to a corresponding ABAP data type. For VARCHAR fields change the output type to NVARCHAR. |
| | If you want to see what this data looks like, click on the Data Preview button.<br><br>Now click on the Raw Data button. |  ...found.<br>Data Preview<br>Analysis<br>Distinct values<br>Raw Data<br>Available objects<br>Labels axis<br>Filter pattern | You should now see data coming from our Hadoop system. Note that this data is not replicated to our HANA database but is non-materialized data.<br><br>As mentioned in the beginning of the script, we have already done the steps below to make this possible.<br>1) Add a Smart Data Access connection in HANA<br>2) Add a Virtual Table, from the Remote Data Source, in HANA<br>3) Create a Calculation View off of the HANA Virtual Table |

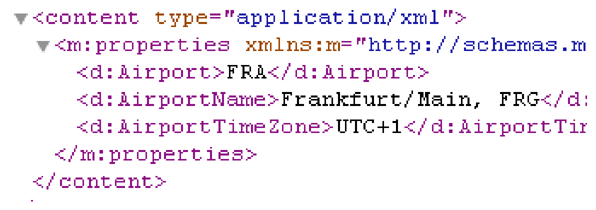
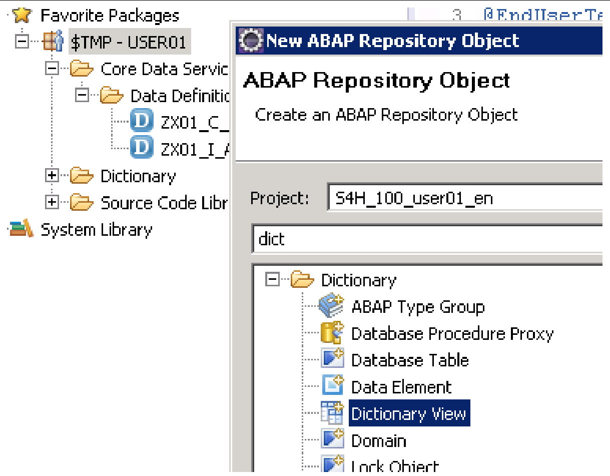
| | What to do | What you should see | Notes |
|---|---|---|---|
| 2 | In Eclipse switch back to the ABAP Perspective.<br><br>In the Project Explorer go to the project you created earlier in the Preparation steps.<br><br>Navigate to:<br>> Favorite Packages<br>> $TMP<br>> CDS<br>> Data Definition<br><br>Open the ZX##_I_Airport CDS view where ## is the last 2 digits of your 0## user name. |  | If you are not yet familiar with CDS, a CDS view although it is developed in the ABAP perspective is not created with ABAP. This type of view consists of:<br><br>1) Annotations that define the view itself (the upper part of the view syntax)<br>2) A SQL statement<br>3) Annotations that further define the record set of the view<br><br>This particular view is described as a BASIC type view as it is an interface view / directly on data tables. It is not meant to be consumed by end user tools but is the lowest level building block of a CDS virtual data model. Multiple BASIC views are combined in COMPOSITE views and the final layer that the end user / developer accesses is the CONSUMPTION type view.<br><br>These annotations may also dictate how products such as analytics behave when they consume these views. For example, some fields may be annotated to be a ROW in analytics or have a default aggregation of MAX. |

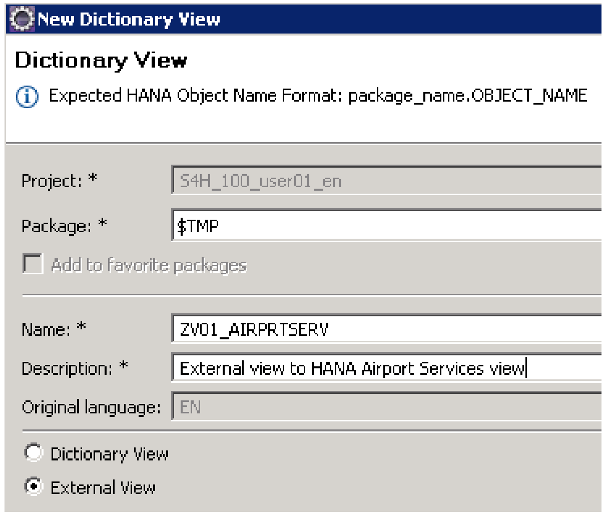
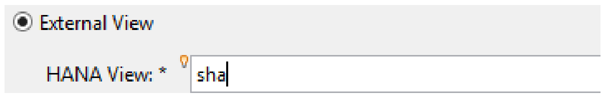
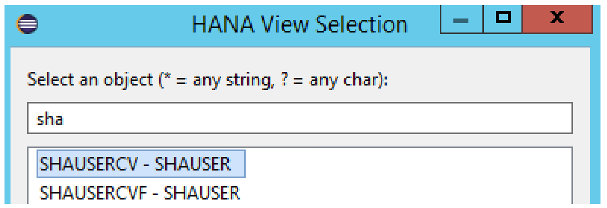
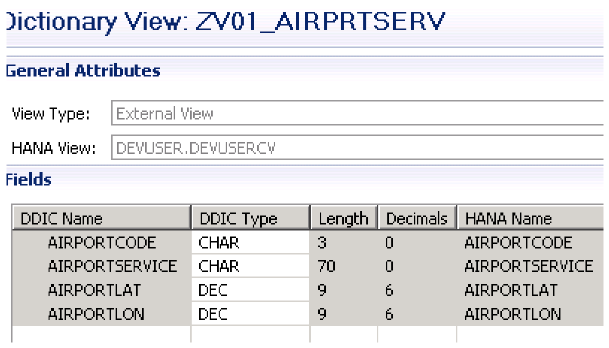
```
S4H_100_user060_en [S4H, 100, USER060, EN]
  Local Objects ($TMP) (7)
    USER060 (7)  sha
      Core Data Services (4)
        Data Definitions (2)
          ZX60_C_AIRPORTQUERY
          ZX60_I_AIRPORT  Airport,
        Entities (2)
      Dictionary (2)
      Source Code Library (1)
```

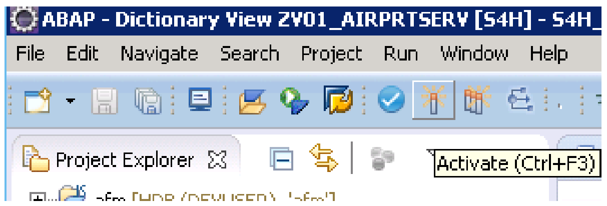
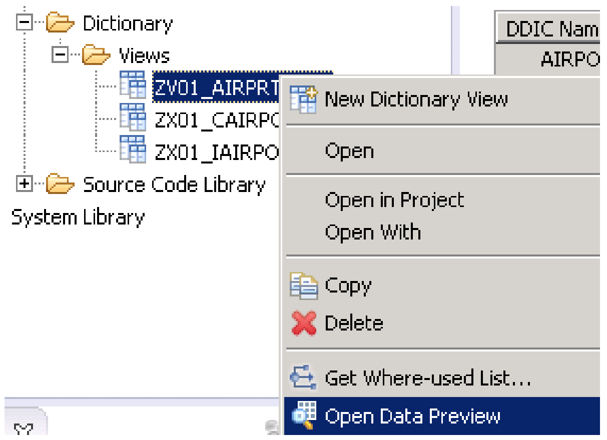
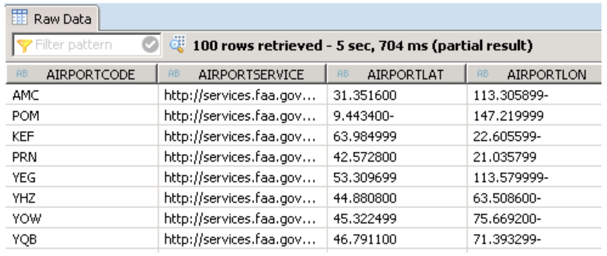
```
@AbapCatalog.sqlViewName: 'ZX6
@AccessControl.authorizationCh
@EndUserText.label: 'Airport'
@Analytics.dataCategory: #DIME
@Analytics.dataExtraction.enab
@VDM.viewType: #BASIC
@ObjectModel.representativeKey

define view ZX60_I_Airport as
select from sairport {
  @ObjectModel.text.element:
```

| | What to do | What you should see | Notes |
|---|---|---|---|
| 3 | Have a look at the code in the top block of this BASIC / Interface CDS view. |  [S4H] - active - S4H_100_user01_en - Eclipse<br>`@AbapCatalog.sqlViewName: 'ZX01_IAIRPORT'`<br>`@AccessControl.authorizationCheck: #NOT_REQUIRED`<br>`@EndUserText.label: 'Airport'`<br>`@Analytics.dataCategory: #DIMENSION`<br>`@Analytics.dataExtraction.enabled: true`<br>`@VDM.viewType: #BASIC`<br>`@ObjectModel.representativeKey: 'Airport'`<br><br>`define view ZX01_I_Airport as`<br>`select from sairport {`<br>`  @ObjectModel.text.element:  [ 'AirportName' ]`<br>`  key sairport.id       as Airport,`<br>`  @Semantics.text`<br>`  sairport.name         as AirportName,`<br>`  sairport.time_zone    as AirportTimeZone`<br>`}`<br><br><br>`@AbapCatalog.sqlViewName: 'ZX01_IAIRPORT'`<br>`@AccessControl.authorizationCheck: #NOT_RE`<br>`@EndUserText.label: 'Airport'`<br>`@Analytics.dataCategory: #DIMENSION`<br>`@Analytics.dataExtraction.enabled: true`<br>`@VDM.viewType: #BASIC`<br>`@ObjectModel.representativeKey: 'Airport'` | When a CDS view is activated 2 views are created:<br>- sqlViewName is the name of the ABAP DDIC view name<br>- the corresponding HANA view uses the "define view" name<br><br>Authorization check Not Required means that no corresponding user security file is used with this view. This data is dimensional data that does not require user level security.<br><br>The end user text label used to describe view later in analytics and for CDS developers.<br><br>The data category dictates to the CDS engine and analytics what the view is used for. Options include fact, dimension, and cube. This view is a dimension type as the fields are descriptive.<br><br>This view has been flagged for later consumption in ETL tools with the data extraction enabled annotation.<br><br>The view type is “basic” as it is directly based on tables / part of the data table interface layer.<br><br>A view can have multiple keys but the representative key is the most relevant or definitive key of the view. |
| 4 | Have a look at the code in the bottom “select” block of this view. | <br>`define view ZX01_I_Airport as`<br>`select from sairport {`<br>`  @ObjectModel.text.element:  [ 'AirportName`<br>`  key sairport.id       as Airport,`<br>`  @Semantics.text`<br>`  sairport.name         as AirportName,`<br>`  sairport.time_zone    as AirportTimeZone`<br>`}`<br><br><br>`key sairport.id      as Airport,`<br>`@Semantics.text`<br>`sairport.na`<br>`sairport.ti`<br>`}`<br>address - annotation<br>address.city: true - annotatic<br>address.country: true - anno | The airport ID is the only key field in the view.<br><br>The text.element annotation is used to assign the corresponding text field to an ID. This is important if using ID’s and text descriptions later for prompts / filters in analytics.<br><br>Airport name is further defined with @semantics annotation as a text field. Highlight "text" and press Control + Space to see many different semantic definitions.<br><br>Note that Control + Space is used for code completion throughout CDS development. This makes coding CDS views much easier. |

|  | What to do | What you should see | Notes |
|---|---|---|---|
| 5 | Now open the ZX##_C_Airport Query CDS view from the Data Definitions. | <br>`@AbapCatalog.sqlViewName: 'ZX01_CAIRPORTQ'`<br>`@AccessControl.authorizationCheck: #NOT_REQU`<br>`@EndUserText.label: 'Airport query'`<br>`@OData.publish: true`<br>`@VDM.viewType: #CONSUMPTION`<br><br>`define view ZX01_C_AirportQuery as`<br>`select from ZX01_I_Airport {`<br>`    key ZX01_I_Airport.Airport,`<br>`    ZX01_I_Airport.AirportName,`<br>`    ZX01_I_Airport.AirportTimeZone`<br>`}` | Note one difference in this view is the viewType of “consumption”. This view is part of the public layer of the virtual data model. In particular this view is available as an odata service via the odata.publish line.<br><br>In the field level notations, note that the "select from" is from the previous airport basic view.<br><br>Normally a virtual data model would have many basic views and also include composite views. However, this is a simple 2 view model built just for demonstration purposes. |
| 6 | In the bottom left of Eclipse there should have an Outline panel.<br><br>In Secondary Objects there is an oData Exposure item.<br><br>Right click on this and choose Open. | <br> Outline<br>ZX01_C_AirportQuery<br>select<br>from<br>select list<br>secondary objects<br>OData Exposure<br>Open<br>Copy<br>Get Where-used List…<br>Rename…<br>Run As<br>Debug As<br>Profile As | You will probably get a warning about the connection not being private. If so press Advanced and then the link to Proceed.<br><br>If you are prompted to choose a browser, choose Chrome and do not choose Internet Explorer…you’ll be looking at web service data which works best in Chrome.<br><br>You will need to log on as user## (where ## is the number you have been assigned) and the password of SHALive1. |
|  | Change the suffice of the url from<br>?sap-ds-debug=true<br>to<br>$metadata<br>& press Enter<br><br>Copy your EntitySet Name…e.g. ZX##_C_Airport… | <br>`<Schema xmlns="http://schemas.microsoft.com/ad`<br>`<EntityType Name="ZX01_C_AirportQueryType" sa`<br>`<Key>`<br>`<PropertyRef Name="Airport"/>`<br>`</Key>`<br>`<Property Name="Airport" Type="Edm.String"`<br>`<Property Name="AirportName" Type="Edm.Stri`<br>`<Property Name="AirportTimeZone" Type="Edm.`<br>`</EntityType>`<br>`<EntityContainer Name="sap.com.cds_zx01_c_air`<br>`<EntitySet Name="ZX01_C_AirportQuery" Entit`<br>`sap:deletable="false" sap:content-version="`<br>`</EntityContainer>`<br>`<atom:link xmlns:atom="http://www.w3.org/2005`<br>`<atom:link xmlns:atom="http://www.w3.org/2005`<br>`href="https://vhcals4hci.dummy.nodomain:4430C` | Changing the suffix to $metadata allows you to see the entity set / fields that the oData service will return. Note that these fields are from the consumption view that exists in the ABAP data dictionary.<br><br>Normally you would not do this (for performance reasons) as you’re returning the full data set using a web service…however in our case we know that there aren’t that many records. |

| | What to do | What you should see | Notes |
|---|---|---|---|
| | After copying the entity set name, replace $metadata with that name. Press Enter. |  ▾`<content type="application/xml">` ▾`<m:properties xmlns:m="http://schemas.m` `<d:Airport>FRA</d:Airport>` `<d:AirportName>Frankfurt/Main, FRG</d:` `<d:AirportTimeZone>UTC+1</d:AirportTir` `</m:properties>` `</content>` | You should now see the data being returned from the S4 system for the Airport Query consumption view.<br><br>Don't close this browser window as we'll use it again.<br><br>In addition to oData, consumption views can also create analytic queries. This workflow will not be covered in this hands-on though. These analytic queries can be consumed in tools such as OAnalysis, Crystal Reports, Analysis for Office, etc. |
| 7 | In the next steps we'll create an External View in the ABAP data dictionary.<br><br>Right click on the $TMP folder and choose New > Other ABAP Repo Object<br><br>Start typing in "dictionary" & select Dictionary View and click Next. |  Favorite Packages, $TMP - USER01, Core Data Servic, Data Definitic, ZX01_C_, ZX01_I_, Dictionary, Source Code Libr, System Library. New ABAP Repository Object — ABAP Repository Object — Create an ABAP Repository Object. Project: S4H_100_user01_en. dict. Dictionary: ABAP Type Group, Database Procedure Proxy, Database Table, Data Element, Dictionary View, Domain, Lock Object | This External View will connect to the HANA Calculation View that we looked at earlier. By adding this External View we will then have connectivity to the Hadoop data source within our ABAP data dictionary / DDIC. |

| | What to do | What you should see | Notes |
|---|---|---|---|
| | Change view type from Dictionary View to External View. | <br>New Dictionary View<br>Dictionary View<br>Expected HANA Object Name Format: package_name.OBJECT_NAME<br>Project: * S4H_100_user01_en<br>Package: * $TMP<br>Add to favorite packages<br>Name: * ZV01_AIRPRTSERV<br>Description: * External view to HANA Airport Services view<br>Original language: EN<br>○ Dictionary View<br>● External View | The following will be the name and description to assign to the view.<br><br>Note that ## must be replaced with your user number. Ensure that your view name also starts with ZV, where V is for view. The Z means that the view will be part of the custom development space as opposed to objects which are shipped with the S4 product. This workflow ensures that we do not directly alter any existing S4 objects.<br><br>Name: ZV##_AIRPRTSERV<br><br>Description: External view to HANA Airport Services view |
| 8 | In HANA View, type in "sha" and press Browse.<br><br>Click on the SHAUSERCV calculation view and press OK.<br><br>Press Finish. | <br>● External View<br>HANA View: * sha<br><br>HANA View Selection<br>Select an object (* = any string, ? = any char):<br>sha<br>SHAUSERCV - SHAUSER<br>SHAUSERCVF - SHAUSER<br><br>Dictionary View: ZV01_AIRPRTSERV<br>General Attributes<br>View Type: External View<br>HANA View: DEVUSER.DEVUSERCV<br>Fields<br>DDIC Name \| DDIC Type \| Length \| Decimals \| HANA Name<br>AIRPORTCODE \| CHAR \| 3 \| 0 \| AIRPORTCODE<br>AIRPORTSERVICE \| CHAR \| 70 \| 0 \| AIRPORTSERVICE<br>AIRPORTLAT \| DEC \| 9 \| 6 \| AIRPORTLAT<br>AIRPORTLON \| DEC \| 9 \| 6 \| AIRPORTLON | We will use the pre-built HANA Calculation View that we looked at earlier. By the way, the other view SHAUSERCVF is from a file system connection. You can add data from files to your S4 system, again using Smart Data Access.<br><br>After pressing Finish you should see the External View that has been created based on the HANA Calculation View.<br><br>There isn't much that can be changed at this stage other than changing some of the field types in the DDIC Type column. This is not necessary in our case.<br><br>Note that DDIC Types have been mapped based on the HANA Types. In general, if there are any errors at this point they may be due the HANA Type not having a corresponding ABAP DDIC Type. |

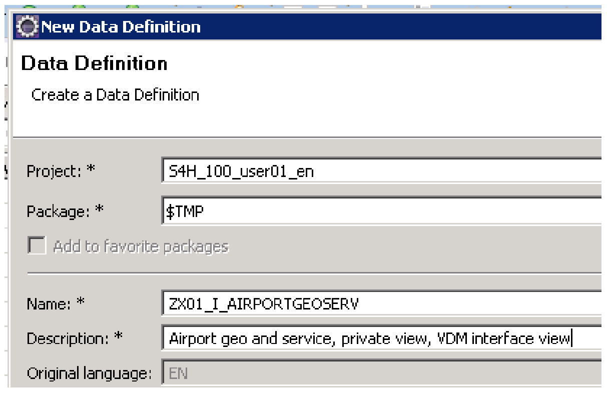
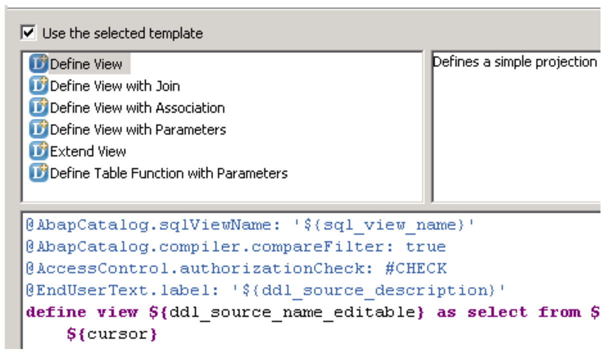
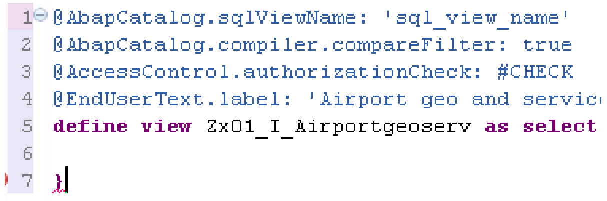
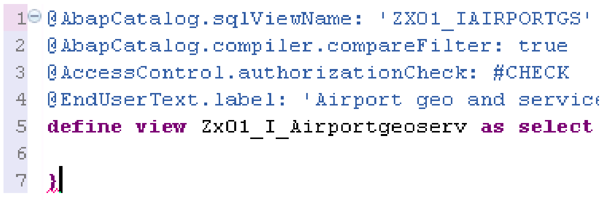
| | What to do | What you should see | Notes |
|---|---|---|---|
| 9 | Now press the Activate button. |  | The view must be activated in order to have the view and fields fully available in CDS. |
| | In the Project Explorer open the Dictionary folder, then Views, then right click on ZV##_AIRPRTSERV view and choose Open With > Data Preview. |  | You should now see the record set returned from the External View. You now have access to our Hadoop data in your ABAP DDIC. |
| 10 | The next major step is to create a BASIC CDS view that will return data from our external view. |  | The reason to create a CDS view off of the ABAP External View is to make the elements of the CDS view available throughout the virtual data model that has already been started in your ABAP project. |

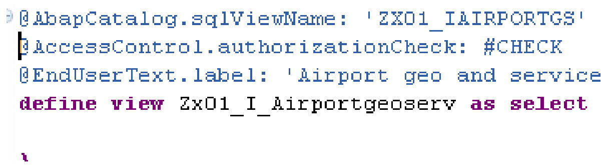
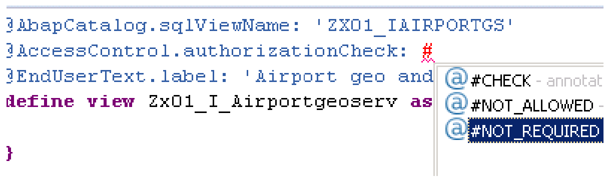
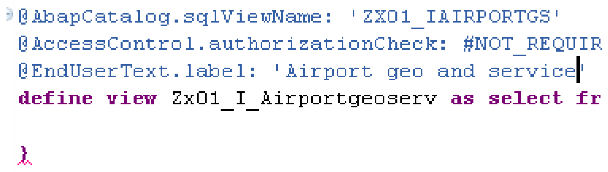
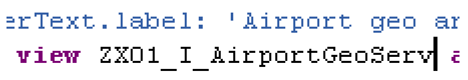
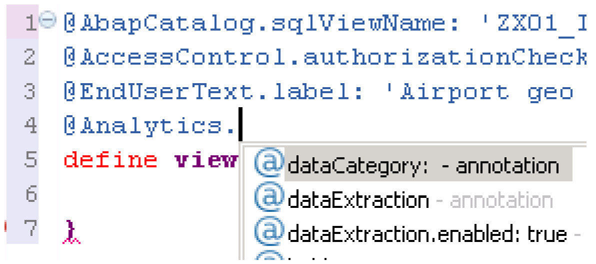
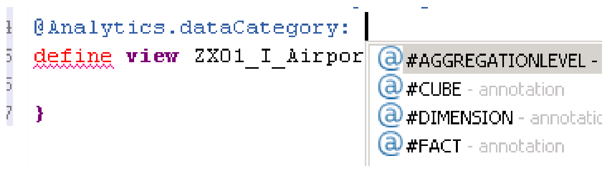
ABAP - Dictionary View ZV01_AIRPRTSERV [S4H] - S4H_
File Edit Navigate Search Project Run Window Help
Project Explorer
Activate (Ctrl+F3)

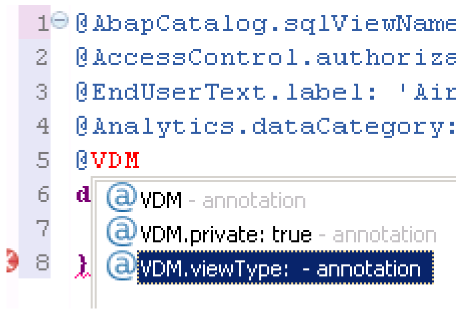
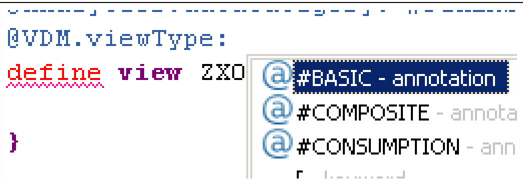
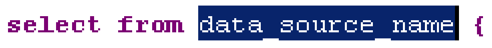
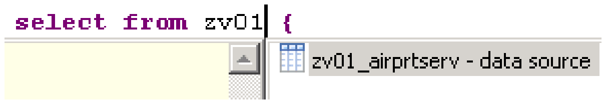
Raw Data

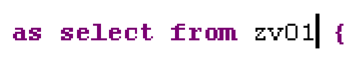
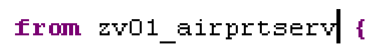
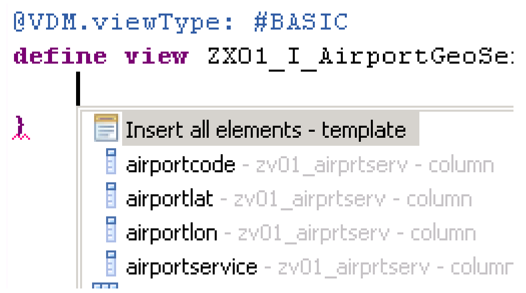
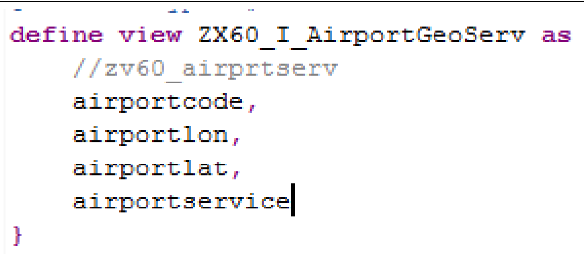
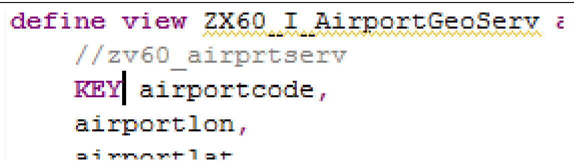
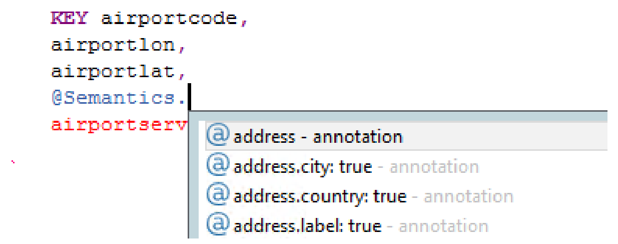
100 rows retrieved - 5 sec, 704 ms (partial result)

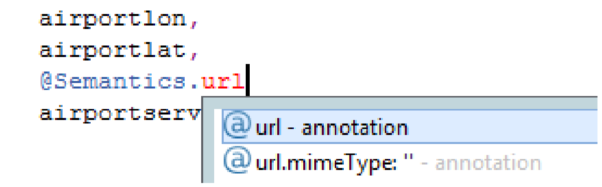
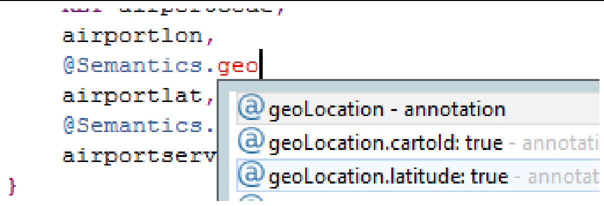
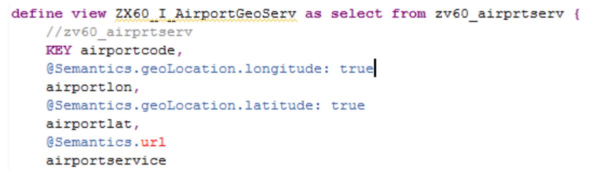
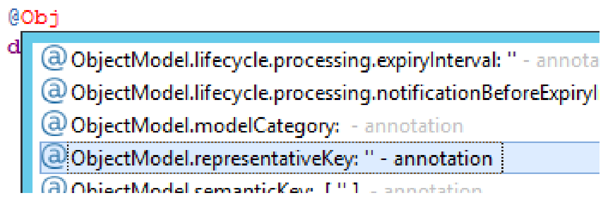
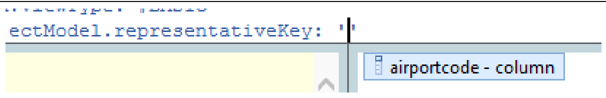
| AIRPORTCODE | AIRPORTSERVICE | AIRPORTLAT | AIRPORTLON |
|---|---|---|---|
| AMC | http://services.faa.gov... | 31.351600 | 113.305899- |
| POM | http://services.faa.gov... | 9.443400- | 147.219999 |
| KEF | http://services.faa.gov... | 63.984999 | 22.605599- |
| PRN | http://services.faa.gov... | 42.572800 | 21.035799 |
| YEG | http://services.faa.gov... | 53.309699 | 113.579999- |
| YHZ | http://services.faa.gov... | 44.880800 | 63.508600- |
| YOW | http://services.faa.gov... | 45.322499 | 75.669200- |
| YQB | http://services.faa.gov... | 46.791100 | 71.393299- |

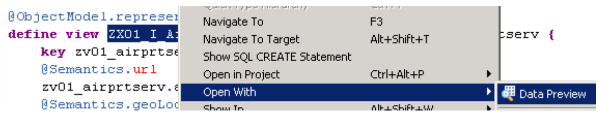
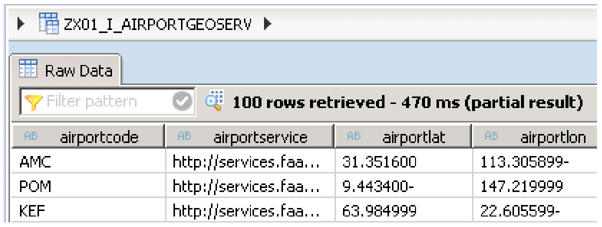
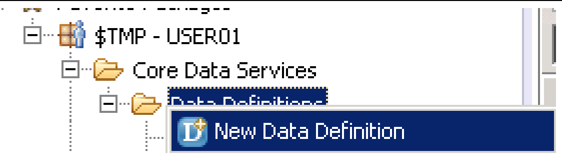
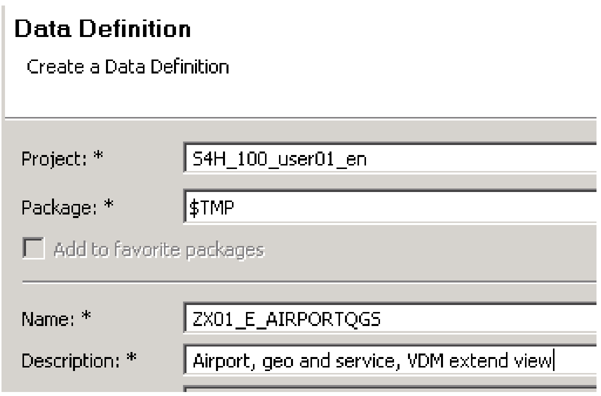
| | What to do | What you should see | Notes |
|---|---|---|---|
| 11 | In the Project Explorer go to $TMP > Core Data Services and right click on Data Definitions.<br><br>Choose New Data Definition. | <br>New Data Definition<br>**Data Definition**<br>Create a Data Definition<br>Project: * S4H_100_user01_en<br>Package: * $TMP<br>☐ Add to favorite packages<br>Name: * ZX01_I_AIRPORTGEOSERV<br>Description: * Airport geo and service, private view, VDM interface view<br>Original language: EN | The following will be the name and description to assign to the view. Note that ## must be replaced with your user number.<br><br>Name: ZX##_I_AirportGeoServ<br><br>Description: Airport geo and service, private view, VDM interface view |
| 12 | Click Next and Next again.<br><br>Ensure that the default of “Define View” is selected. | <br>☑ Use the selected template<br>Define View<br>Define View with Join<br>Define View with Association<br>Define View with Parameters<br>Extend View<br>Define Table Function with Parameters<br>Defines a simple projection<br>`@AbapCatalog.sqlViewName: '${sql_view_name}'`<br>`@AbapCatalog.compiler.compareFilter: true`<br>`@AccessControl.authorizationCheck: #CHECK`<br>`@EndUserText.label: '${ddl_source_description}'`<br>`define view ${ddl_source_name_editable} as select from $`<br>`${cursor}` | Note that there are several templates available which provide some basic code to help you create your CDS view.<br><br>You can see some of the basic code that the template has provided and some of the annotations we saw in the previous views. |
| 13 | Press Finish.<br><br>In the next steps we will write a basic type CDS view. | <br>`1 @AbapCatalog.sqlViewName: 'sql_view_name'`<br>`2 @AbapCatalog.compiler.compareFilter: true`<br>`3 @AccessControl.authorizationCheck: #CHECK`<br>`4 @EndUserText.label: 'Airport geo and servic`<br>`5 define view Zx01_I_Airportgeoserv as select`<br>`6`<br>`7 }` | Note that there is default syntax written out for a simple view. We’ve seen most of these annotations before when we reviewed the existing views.<br><br>Developing in CDS will be fairly easy as there is code completion (Ctrl + Space) and there are only a certain number of annotations that you need to remember to add for simple CDS views. |
| 14 | On line number 1, change `'sql_view_name'` to 'ZX##_IAIRPORTGS' replacing ## with your user number. | <br>`1 @AbapCatalog.sqlViewName: 'ZX01_IAIRPORTGS'`<br>`2 @AbapCatalog.compiler.compareFilter: true`<br>`3 @AccessControl.authorizationCheck: #CHECK`<br>`4 @EndUserText.label: 'Airport geo and servic`<br>`5 define view Zx01_I_Airportgeoserv as select`<br>`6`<br>`7 }` | Note that the sqlViewName is slightly different (the underscore after “_I” has been removed) than the name that we are going to give to the DDL file name / define view name. This view name should be all uppercase.<br><br>This is because there are 2 different objects created by the activation process:<br>1) An ABAP DDIC view name (the sqlViewName)<br>2) A HANA SQL view |

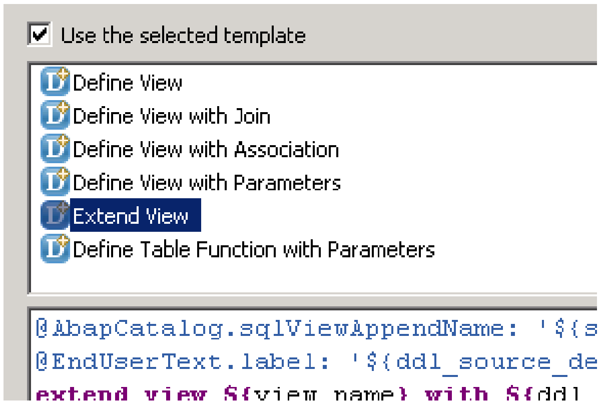
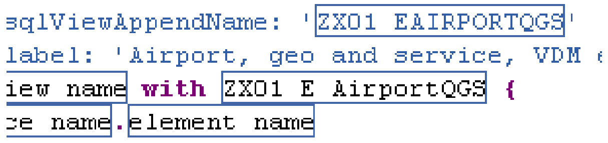
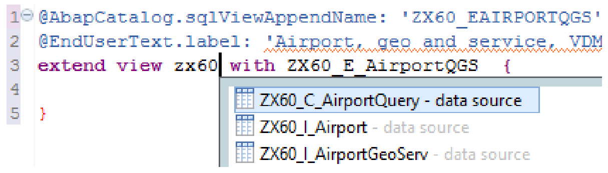
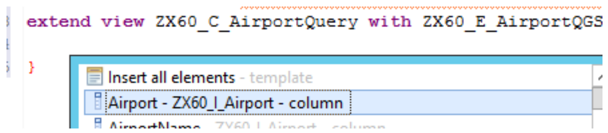
| | What to do | What you should see | Notes |
|---|---|---|---|
| 15 | You can delete line number 2 with the compareFilter annotation. |  @AbapCatalog.sqlViewName: 'ZX01_IAIRPORTGS'<br>@AccessControl.authorizationCheck: #CHECK<br>@EndUserText.label: 'Airport geo and service<br>define view Zx01_I_Airportgeoserv as select | This particular annotation is used when a CDS view has more than 1 filter and you want to optimize the data filtering process by comparing like filters. As this view has only 1 select clause and no actual filter annotation, we can remove the line. Setting "true" to "false" also has the same effect. |
| 16 | In the authorization check line, delete "CHECK" from the end of the line,<br>press Ctrl + Space,<br>then select NOT_REQUIRED. |  @AbapCatalog.sqlViewName: 'ZX01_IAIRPORTGS'<br>@AccessControl.authorizationCheck: #<br>@EndUserText.label: 'Airport geo and<br>define view Zx01_I_Airportgeoserv as<br>}<br> | We do not need to check for a corresponding security file (DCL file) as we are not using user based security on this data. This data is from an external data source and is just dimensional / descriptive data that does not need additional security.<br><br>This particular annotation prevents an error message from being displayed when there is no security file in place. |
| 17 | The End User Text label can be shortened to 'Airport geo and service'. |  @AbapCatalog.sqlViewName: 'ZX01_IAIRPORTGS'<br>@AccessControl.authorizationCheck: #NOT_REQUIR<br>@EndUserText.label: 'Airport geo and service'<br>define view Zx01_I_Airportgeoserv as select fr<br>} | The end user text label is a description that is used so that CDS developers and analytic developers have an idea of what the view does / contains for data. |
| 18 | The "define view" name should be using the proper naming convention. |  erText.label: 'Airport geo ar<br>view ZX01_I_AirportGeoServ a | CDS view names should use the naming convention / case such as you see in this picture. The first 2 letters of the name should be in capitals, the view type (I is for interface view) should be capitalized, and the remaining part of the name should use upper case for the start of each new 'word'. |
| 19 | Add a new line after the end user text label.<br><br>Type in "@Analytics." (with a period at the end) and you'll see code completion for this annotation. |  @AbapCatalog.sqlViewName: 'ZX01_I<br>@AccessControl.authorizationCheck<br>@EndUserText.label: 'Airport geo<br>@Analytics.<br>define view<br>}<br> dataCategory: - annotation<br>dataExtraction - annotation<br>dataExtraction.enabled: true - | A basic level CDS view should have a data category. This data category is propagated through the virtual data model. This is further described in the next step. |
| 20 | Select dataCategory and then press Ctrl (control) + Space for further code completion. |  @Analytics.dataCategory:<br>define view ZX01_I_Airpor<br>}<br> #AGGREGATIONLEVEL -<br>#CUBE - annotation<br>#DIMENSION - annotatic<br>#FACT - annotation | These data categories are used to describe the data when analytics are used later. For example some analytics will automatically have access to any cube type views. Any dimension or fact type views can be used to define a cube in a later view and of course those fields will be marked with those fact or dimension categories. |

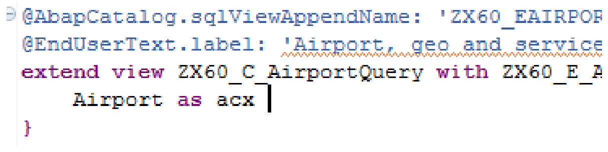
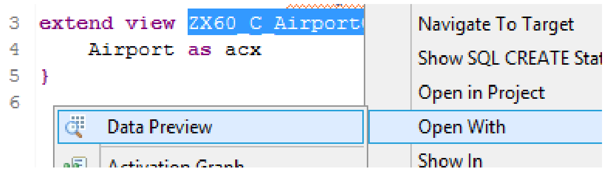
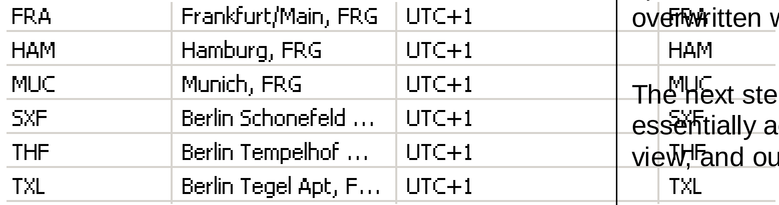
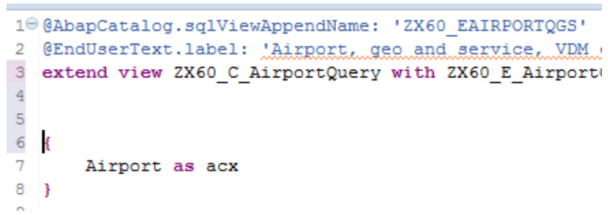
| | What to do | What you should see | Notes |
|---|---|---|---|
| 21 | Select #DIMENSION for this view.<br><br>Add another line below dataCategory and type in "@VDM" and then select @VDM.viewType. | <br>`@AbapCatalog.sqlViewName`<br>`@AccessControl.authorize`<br>`@EndUserText.label: 'Air`<br>`@Analytics.dataCategory:`<br>`@VDM`<br>`d`<br>`}`<br>VDM - annotation<br>VDM.private: true - annotation<br>VDM.viewType: - annotation | CDS views should in most cases have a view type which defines where it fits into the overall virtual data model. This is further described in the next step. |
| | Press Ctrl + Space and then select #BASIC. | <br>`@VDM.viewType:`<br>`define view ZXO`<br>`}`<br>#BASIC - annotation<br>#COMPOSITE - annota<br>#CONSUMPTION - ann | This is a basic CDS view as it is against an External View so we can treat it as part of the interface level.<br><br>Basic / interface views are the lower building blocks on the CDS virtual data model 'pyramid' with composite views in the middle which combine many basic views, and the consumption views on the top which are exposed to the analytics developer or to end user applications via oData or analytic queries or HANA SQL views. |
| | For now we are done with the view level annotations and need to start defining the view's fields / entities.<br><br>Highlight and delete data_source_name and then type in ZV## where ## is your user number.<br><br>Press Ctrl + Space and try to select your view. If that does not work see the next step. | <br>`select from data_source_name {`<br><br><br>`select from zv01 {`<br>zv01_airprtserv - data source | When you type in ZV## you'll see a list of available ABAP DDIC data sources in your development space. |

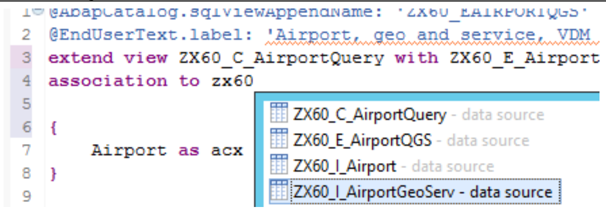
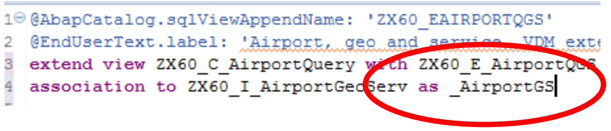
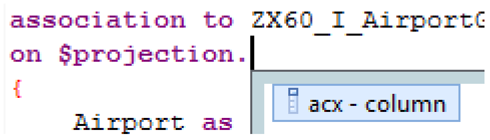
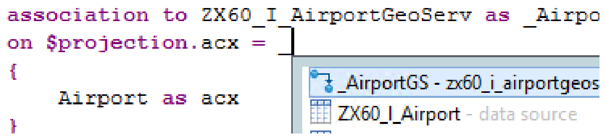
| | What to do | What you should see | Notes |
|---|---|---|---|
| | If you are not able to select the view then type in the name manually…it should be similar to<br>zv##_airprtserv |  `as select from zv01 {`<br> `from zv01_airprtserv {` | Normally you can select these data sources but in Eclipse you may see an issue at this point with selecting existing data sources. |
| | Place your cursor in the line below the "define view" line and then press Ctrl + Space. |  `@VDM.viewType: #BASIC`<br>`define view ZX01_I_AirportGeoSe`<br>`}`<br>Insert all elements - template<br>airportcode - zv01_airprtserv - column<br>airportlat - zv01_airprtserv - column<br>airportlon - zv01_airprtserv - column<br>airportservice - zv01_airprtserv - column | You should see all of the fields that are in the External View that is in your ABAP data dictionary.<br><br>If you do not, check to see that your view name in the "select from" clause is correct. If this name is correct, ensure that you activated the External View that you created earlier. |
| | Choose "Insert all elements". |  `define view ZX60_I_AirportGeoServ as`<br>`    //zv60_airprtserv`<br>`    airportcode,`<br>`    airportlon,`<br>`    airportlat,`<br>`    airportservice`<br>`}` | This will add all available fields from the External View into our basic level CDS view.<br><br>The next step is to further define the view using field level annotations. |
| | The first step is to assign the main view fields as KEY fields.<br><br>Add "KEY" to the airportcode line. |  `define view ZX60_I_AirportGeoServ a`<br>`    //zv60_airprtserv`<br>`    KEY airportcode,`<br>`    airportlon,` | This view has only one KEY field which is the 3-letter airport code.<br><br>The next steps is to add semantic annotations to some of the view's fields to further define them for analytics or applications. Some of these semantic annotations will dictate how the fields are formatted in these end user tools. |
| | Add a new line above the airportservice line.<br><br>Type in "@Semantics." with a period. |  `KEY airportcode,`<br>`airportlon,`<br>`airportlat,`<br>`@Semantics.`<br>`airportserv`<br>address - annotation<br>address.city: true - annotation<br>address.country: true - annotation<br>address.label: true - annotation | Here you can see all of the available semantics options that we looked at earlier in one of the existing CDS views. |

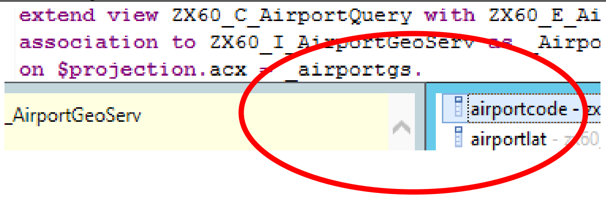
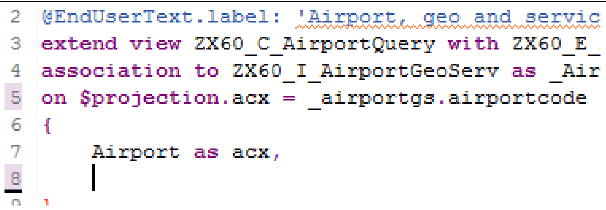
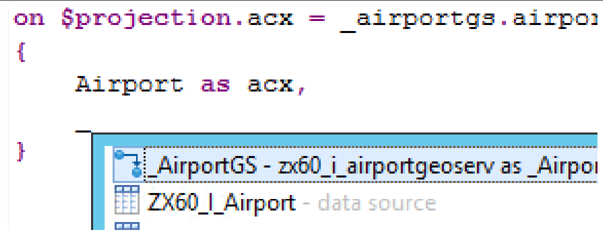
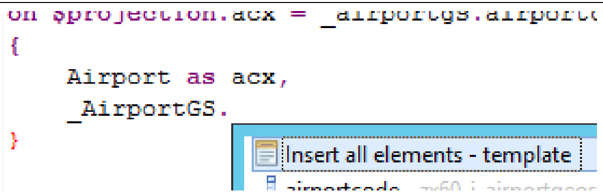
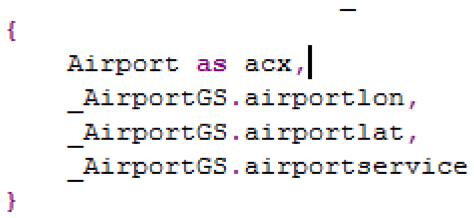
| | What to do | What you should see | Notes |
|---|---|---|---|
| | Start to type in “url” and then select the “url” semantic. |  `airportlon,`<br>`airportlat,`<br>`@Semantics.url`<br>`airportserv`<br>url - annotation<br>url.mimeType: '' - annotation | The airport service field is a URL for a web service that returns available current information for an airport. Note that not all airports have this type of web service. |
| | Insert a line above the airportlat field. Type in “@Semantics.” and then start to type in “geo”. Select the geoLocation latitude semantic. |  `airportlon,`<br>`@Semantics.geo`<br>`airportlat,`<br>`@Semantics.`<br>`airportserv`<br>`}`<br>geoLocation - annotation<br>geoLocation.cartoId: true - annotati<br>geoLocation.latitude: true - annotat | |
| | Repeat the above step but for the longitude field. |  `define view ZX60_I_AirportGeoServ as select from zv60_airprtserv {`<br>`    //zv60_airprtserv`<br>`    KEY airportcode,`<br>`    @Semantics.geoLocation.longitude: true`<br>`    airportlon,`<br>`    @Semantics.geoLocation.latitude: true`<br>`    airportlat,`<br>`    @Semantics.url`<br>`    airportservice` | We are now done defining the field level annotations of our CDS view.<br><br>There is one more step to do in the view level annotations. |
| | Add a line below the VDM viewType in the view level annotations.<br><br>Type in “@Obj” and scroll down and then select the representative key annotation. |  `@Obj`<br>ObjectModel.lifecycle.processing.expiryInterval: '' - annota<br>ObjectModel.lifecycle.processing.notificationBeforeExpiryI<br>ObjectModel.modelCategory: - annotation<br>ObjectModel.representativeKey: '' - annotation | Most CDS views will have a key field that represents the view more than the other keys. In this case we have only one key field so that choice will be easy. |
| | Press Ctrl + Space and select 'airportcode'. |  `ectModel.representativeKey: ''`<br>airportcode - column | We are now done coding the airport geo and service CDS view. |

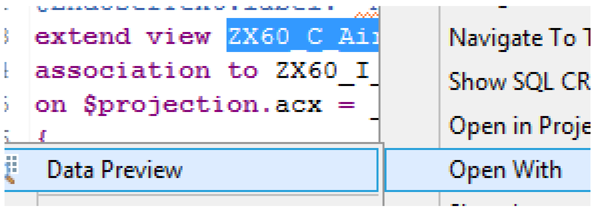
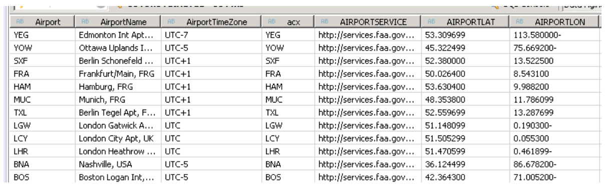
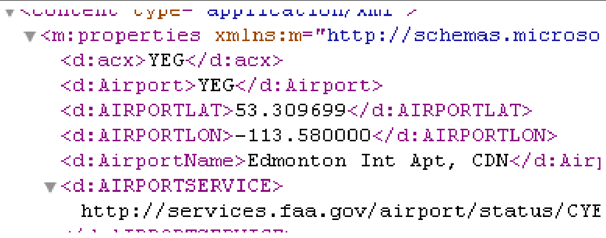
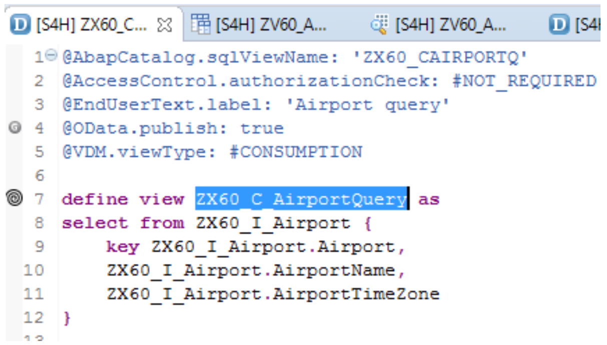
| | What to do | What you should see | Notes |
|---|---|---|---|
| | Press the Activate button or Ctrl + F3.<br><br>Double click on your view name and choose Open With and then Data Preview. | <br>`@ObjectModel.represe`<br>`define view ZX01_I_A` … `tserv {`<br>`key zv01_airprtse`<br>`@Semantics.url`<br>`zv01_airprtserv.a`<br>`@Semantics.geoLoc`<br> Navigate To F3<br>Navigate To Target Alt+Shift+T<br>Show SQL CREATE Statement<br>Open in Project Ctrl+Alt+P<br>Open With<br>Data Preview<br>Show In Alt+Shift+W<br><br> <br>100 rows retrieved - 470 ms (partial result)<br>airportcode \| airportservice \| airportlat \| airportlon<br>AMC \| http://services.faa... \| 31.351600 \| 113.305899-<br>POM \| http://services.faa... \| 9.443400- \| 147.219999<br>KEF \| http://services.faa... \| 63.984999 \| 22.605599- | You should now see data returned from your CDS view.<br><br>We have now gone from:<br>- A Hadoop data source to<br>- A Smart Data Access connection in HANA to<br>- A Remote Data Source Table in HANA to<br>- A HANA Calculation View to<br>- An ABAP DDIC External View to<br>- An ABAP DDIC basic CDS view<br><br>The next step is to extend an existing view in our virtual data model with an Extend View in CDS. This will be further described in the next steps. |
| | Right click on your Data Definitions folder and choose New Data Definition. | <br> $TMP - USER01<br>Core Data Services<br>Data Definitions<br>New Data Definition | |
| | Give it a name like<br><br>ZX##_E_AirportQGS<br><br>Use the description that is in the Notes to the right >> | <br>**Data Definition**<br>Create a Data Definition<br>Project: * S4H_100_user01_en<br>Package: * $TMP<br>☐ Add to favorite packages<br>Name: * ZX01_E_AIRPORTQGS<br>Description: * Airport, geo and service, VDM extend view | Don't forget to change ## to your user number.<br><br>The description can be something like this: Airport, geo and service, VDM extend view<br><br>The "E" in this case will refer to an Extend View…more about this in the next steps. |

| | What to do | What you should see | Notes |
|---|---|---|---|
| | Press Next and then Next again.<br><br>This time change the template to Extend View.<br><br>Press Finish after looking at the sample code in the template. | ☑ Use the selected template<br>Define View<br>Define View with Join<br>Define View with Association<br>Define View with Parameters<br>Extend View<br>Define Table Function with Parameters<br>`@AbapCatalog.sqlViewAppendName: '${s`<br>`@EndUserText.label: '${ddl_source_de`<br>`extend view ${view_name} with ${ddl` | This time we are going to extend an existing CDS view. In particular we will extend the existing consumption view that we looked at earlier.<br><br>Using the Extend View workflow, we'll add fields from our new basic level airport geo service view to our consumption view, without having to physically modify the target consumption view.<br><br>This is useful when you want to add fields to an existing view without having to alter that view. For example, you should not physically change any CDS views that ship with the S4 product or those changes can be overwritten when you update your system. By using an Extend View CDS that is in your personal development space, you are not actually physically changing the target view. |
| | Make two name changes to your view according to the Notes on the right >> | `sqlViewAppendName: 'ZX01_EAIRPORTQGS'`<br>`label: 'Airport, geo and service, VDM`<br>`iew_name with ZX01_E_AirportQGS {`<br>`ce_name.element_name` | On the first line of the default code, change the sqlViewAppendView to ZX##_EAIRPORTQGS substituting ## for your user number.<br><br>Change the "with view name" to have the proper case for CDS like `ZX##_E_AirportQGS` |
| | In the "extend view view_name" line, change "view_name" to "zx##" using your user number and press Ctrl + Space.<br><br>Select the AirportQuery consumption view. | `@AbapCatalog.sqlViewAppendName: 'ZX60_EAIRPORTQGS'`<br>`@EndUserText.label: 'Airport, geo and service, VDM`<br>`extend view zx60 with ZX60_E_AirportQGS {`<br>`}`<br>ZX60_C_AirportQuery - data source<br>ZX60_I_Airport - data source<br>ZX60_I_AirportGeoServ - data source | You should now see 3 CDS views from your development space.<br><br>We are extending the ZX##_C_AitrportQuery consumption view so that it will include the fields from the basic view that contains our airport web service and the airport latitude and longitude fields |
| | Delete the 4th line of `data_source_name.element_name` and press Ctrl + Space.<br><br>This time select only the Airport column. | `extend view ZX60_C_AirportQuery with ZX60_E_AirportQGS`<br>`}`<br>Insert all elements - template<br>Airport - ZX60_I_Airport - column<br>AirportName - ZX60_I_Airport - column | Do not select "Insert all elements" this time as we are only going to need the Airport field, which is the 3 letter airport ID.<br><br>The fields that you see are actually in the existing consumption view. We want to later add the fields from our airport geo and service view though.<br><br>We do need the consumption view's Airport field here first though as we're going to use it as a link to our airport service and geo view. |

| | What to do | What you should see | Notes |
|---|---|---|---|
| | We need to provide an alias for this “id” field so change the 4th line to have an alias of “acx”. |  `@AbapCatalog.sqlViewAppendName: 'ZX60_EAIRPOR`<br>`@EndUserText.label: 'Airport, geo and service`<br>`extend view ZX60_C_AirportQuery with ZX60_E_A`<br>`    Airport as acx`<br>`}` | We need to change the 4th line to `Airport as acx` so that an alias (“acx” for airport code, extend) is used. The existing consumption view already has the airport code in it so when we extend it, we don’t want any duplicate field names.<br><br>We’re also adding another view later on, the airport geo and service view, and that view will have an airport code field that we be ‘joining’ on using an association. |
| | Activate this view and then select the view name.<br><br>Right click and choose Open With and then choose Data Preview. |  `extend view ZX60_C_Airport`<br>`    Airport as acx`<br>`}`<br><br> FRA \| Frankfurt/Main, FRG \| UTC+1 \| FRA<br>HAM \| Hamburg, FRG \| UTC+1 \| HAM<br>MUC \| Munich, FRG \| UTC+1 \| MUC<br>SXF \| Berlin Schonefeld … \| UTC+1 \| SXF<br>THF \| Berlin Tempelhof … \| UTC+1 \| THF<br>TXL \| Berlin Tegel Apt, F… \| UTC+1 \| TXL | Note that we’re actually looking at the extended airport query consumption view which has our airport code / alias column appended to its record set.<br><br>The original airport query consumption view has not been physically changed and since we’ve used this Extend View workflow in our custom development space, we don’t need to worry about any existing data model views being overwritten with system updates.<br><br>The next step is add an association to our Extend View syntax, which essentially acts like a join between the existing airport query consumption view, and our new airport geo and service basic view. |
| | Back in our Extend View code, add two lines after the “extend view” line but before the opening brace. |  `@AbapCatalog.sqlViewAppendName: 'ZX60_EAIRPORTQGS'`<br>`@EndUserText.label: 'Airport, geo and service, VDM`<br>`extend view ZX60_C_AirportQuery with ZX60_E_Airport`<br><br><br>`{`<br>`    Airport as acx`<br>`}` | Your code should now look like this on the left where we have 2 blank lines before the opening brace / bracket of our entity set / selected columns code. |

| | What to do | What you should see | Notes |
|---|---|---|---|
| | In the first blank line (should be line 4) and type in “association to” and then type in “zx##” where ## is your user number.<br><br>Press Ctrl + Space to see available views.<br><br>Select the ZX##_I_ AirportGeoServ view. | <br>`1 @AbapCatalog.sqlViewAppendName: 'ZX60_EAIRPORTQGS'`<br>`2 @EndUserText.label: 'Airport, geo and service, VDM`<br>`3 extend view ZX60_C_AirportQuery with ZX60_E_Airport`<br>`4 association to zx60`<br>`5`<br>`6 {`<br>`7     Airport as acx`<br>`8 }`<br>`9`<br>ZX60_C_AirportQuery - data source<br>ZX60_E_AirportQGS - data source<br>ZX60_I_Airport - data source<br>ZX60_I_AirportGeoServ - data source | We are going to associate this new Extend View to the airport geo service view (Hadoop data) that we built several steps ago.<br><br>By adding this association, it essentially joins the two existing views inside this new Extend View. After the association syntax is completed, we will be able to add the fields from our airport geo service view to this new Extend View. |
| | Type in<br>“as _AirportGS”<br>right after the AirportGeoServ view name. | <br>`1 @AbapCatalog.sqlViewAppendName: 'ZX60_EAIRPORTQGS'`<br>`2 @EndUserText.label: 'Airport, geo and service, VDM exte`<br>`3 extend view ZX60_C_AirportQuery with ZX60_E_AirportQGS`<br>`4 association to ZX60_I_AirportGeoServ as _AirportGS` | We need to provide a name for the association. An association name in CDS will always start with an underscore / “_”. |
| | In the next blank line type in “on $” and press Tab to accept “$projection” as the suggested code.<br><br>Type a period “.” right after $projection and then select the “acx” column. | <br>`association to ZX60_I_AirportG`<br>`on $projection.`<br>`{`<br>`    Airport as`<br>acx - column | The “acx” column was the aliased airport ID column from our airport query consumption view. |
| | Type in “ = _” and then Ctrl + Space.<br><br>Ignore any errors should they occur at this point. | <br>`association to ZX60_I_AirportGeoServ as _Airpo`<br>`on $projection.acx = _`<br>`{`<br>`    Airport as acx`<br>`}`<br>_AirportGS - zx60_i_airportgeos<br>ZX60_I_Airport - data source | By typing in the underscore / “_” code completion will show any available associations created in the virtual data model.<br><br>As the AirportGS association is the first one in our data model, there is only one in the suggested associations list. |

| | What to do | What you should see | Notes |
|---|---|---|---|
| | Select the _AirportGS association.<br><br>Type in a period right after and then select the airportcode field. | <br>`extend view ZX60_C_AirportQuery with ZX60_E_Ai`<br>`association to ZX60_I_AirportGeoServ as _Airpo`<br>`on $projection.acx = _airportgs.`<br> | The association has now created what is essentially a join between 2 of the existing CDS views based on the key airport code field. |
| | Go to the select block and a comma and a new line after<br>"**as** acx" | <br>`@EndUserText.label: 'Airport, geo and servic`<br>`extend view ZX60_C_AirportQuery with ZX60_E_`<br>`association to ZX60_I_AirportGeoServ as _Air`<br>`on $projection.acx = _airportgs.airportcode`<br>`{`<br>`    Airport as acx,` | We can now add on to the elements / columns from our external data into our Extend View. |
| | Type in an underscore in the new line and press Ctrl + Space.<br><br>Select the AirportGS associated view. | <br>`on $projection.acx = _airportgs.airpo`<br>`{`<br>`    Airport as acx,`<br>`    _`<br>`}`<br> _AirportGS - zx60_i_airportgeoserv as _Airpo<br>ZX60_I_Airport - data source | After adding the association we will be able to add fields from the airport geo service view. |
| | Type in a period after _AirportGS and then select "Insert all elements". | <br>`on $projection.acx = _airportgs.airportc`<br>`{`<br>`    Airport as acx,`<br>`    _AirportGS.`<br>`}`<br> | |
| | Delete the line for the airportcode line which was just added. | <br>`{`<br>`    Airport as acx,`<br>`    _AirportGS.airportlon,`<br>`    _AirportGS.airportlat,`<br>`    _AirportGS.airportservice`<br>`}` | We only need the airport web service and the latitude and longitude fields from the external data. We don't need the airportcode field in as that would be redundant in the view. |

| | What to do | What you should see | Notes |
|---|---|---|---|
| | Activate the view. Then select the ZX##_C AirportQuery view name choose Open With > Data Preview. |   | We should now see the data that comes from Hadoop in our Extend View.<br><br>We have now completed the coding necessary to add the external data into our virtual data model. |
| | Go back to the Chrome browser tab that you had open earlier for the oData web service. Press the reload button (perhaps twice) to see the new data in the oData web service. |  | We have now combined existing data in an S4 virtual data model with external data and exposed the combined data in an oData web service. |
| | Go back to the ZX##_C_ AirportQuery view which should be in an open tab in Eclipse. |  | Note that the original view has not been altered in any way.<br><br>If you hover your mouse over the new icon beside the “define view” line though, you’ll see that this view has been extended by our new ZX##_E_AirportQGS extend view. |

```
extend view ZX60_C_Ai
association to ZX60_I_
on $projection.acx = _
{
```

| Airport | AirportName | AirportTimeZone | acx | AIRPORTSERVICE | AIRPORTLAT | AIRPORTLON |
|---|---|---|---|---|---|---|
| YEG | Edmonton Int Apt... | UTC-7 | YEG | http://services.faa.gov... | 53.309699 | 113.580000- |
| YOW | Ottawa Uplands I... | UTC-5 | YOW | http://services.faa.gov... | 45.322499 | 75.669200- |
| SXF | Berlin Schonefeld ... | UTC+1 | SXF | http://services.faa.gov... | 52.380000 | 13.522500 |
| FRA | Frankfurt/Main, FRG | UTC+1 | FRA | http://services.faa.gov... | 50.026400 | 8.543100 |
| HAM | Hamburg, FRG | UTC+1 | HAM | http://services.faa.gov... | 53.630400 | 9.988200 |
| MUC | Munich, FRG | UTC+1 | MUC | http://services.faa.gov... | 48.353800 | 11.786099 |
| TXL | Berlin Tegel Apt, F... | UTC+1 | TXL | http://services.faa.gov... | 52.559699 | 13.287699 |
| LGW | London Gatwick A... | UTC | LGW | http://services.faa.gov... | 51.148099 | 0.190300- |
| LCY | London City Apt, UK | UTC | LCY | http://services.faa.gov... | 51.505299 | 0.055300 |
| LHR | London Heathrow ... | UTC | LHR | http://services.faa.gov... | 51.470599 | 0.461899- |
| BNA | Nashville, USA | UTC-5 | BNA | http://services.faa.gov... | 36.124499 | 86.678200- |
| BOS | Boston Logan Int,... | UTC-5 | BOS | http://services.faa.gov... | 42.364300 | 71.005200- |

```
<m:properties xmlns:m="http://schemas.microso
  <d:acx>YEG</d:acx>
  <d:Airport>YEG</d:Airport>
  <d:AIRPORTLAT>53.309699</d:AIRPORTLAT>
  <d:AIRPORTLON>-113.580000</d:AIRPORTLON>
  <d:AirportName>Edmonton Int Apt, CDN</d:Air
  <d:AIRPORTSERVICE>
    http://services.faa.gov/airport/status/CYE
```

```
@AbapCatalog.sqlViewName: 'ZX60_CAIRPORTQ'
@AccessControl.authorizationCheck: #NOT_REQUIRED
@EndUserText.label: 'Airport query'
@OData.publish: true
@VDM.viewType: #CONSUMPTION

define view ZX60_C_AirportQuery as
select from ZX60_I_Airport {
    key ZX60_I_Airport.Airport,
    ZX60_I_Airport.AirportName,
    ZX60_I_Airport.AirportTimeZone
}
```

| | What to do | What you should see | Notes |
|---|---|---|---|
| 22 | Relax as you're done. | | For lots of videos on lots of topics on SAP S/4HANA, please visit the SAP HANA Academy's site here.<br><br>Note that in this particular work flow, we did not show our combined data in the S4 Launchpad / Smart Business Suite. We have a full video series that does that as well as includes the steps for using Smart Data Access to a Hadoop system or from an external file. |