[Содержание](#Содержание)

# [Core Data Services](#Core_Data_Services)

**ABAP in SAP BW**

# **Как отлаживать ABAP в трансформациях**

<https://epotseluev.com/abap/howto-debug-abap-in-transformations/>

SAP **HANA** (High Performance Analytic A’ppliance (ə'plaɪən(t)s устройство))

**OLTP** - Online Transactional Processing

**OLAP** - Online Analytical Processing

Постолбцовое хранилище во многих случаях устраняет необходимость в индексах. Хранение данных в столбцах функционально аналогично наличию встроенного индекса для каждого столбца. Скорость сканирования столбцов в in-memory хранилище и механизмы сжатия, особенно словарный метод сжатия, позволяют выполнять операции чтения с очень высокой производительностью. Во многих случаях необязательно иметь дополнительные индексы.

**HTC** - это объект разработки ABAP. Он необходим для интеграции содержимого репозитория HANA в стандартную систему изменений и транспорта (**CTS**).

Начиная с AS ABAP 7.4, HTC легко интегрируется в транспортный органайзер AS ABAP и интегрирует содержимое хранилища HANA в CTS. Это обеспечивает эффективный процесс доставки приложений, построенных на ABAP (скажем, метода) и контента HANA (например, AMDP), или просто ABAP для приложений SAP HANA между системами SAP с помощью проверенного транспортного механизма ABAP.

Core Data Services (**CDS**) - механизм для переноса логики в базу данных.

**BEx User Exit -** Allows the creation and population of variables and calculations for  
  key figures and variables on a runtime basis. This user exit is called each  
  time BEx is executed.

**BW Transfer rules -** These rules are invoked as data is transferred from R/3 to BW, or from  
  within BW as export data sources and InfoSources are created.

**BW Update Rules -** These rules are invoked as data is being updated into data targets.  
 Whether ODS Objects or InfoCubes, each update results in the execution of an  
  Update Rule.

**BW Start Routines -** Start routines in BW Update Rules are used to manage whole subsets of  
 data. This is different than a simple update rule in that update rules are  
data element specific.

# **ABAP for SAP HANA’ Points to Remember**

<https://sapyard.com/abap-for-sap-hana-points-to-remember/>

SAP **HANA** - **High Performance Analytic A’ppliance** (ə'plaɪən(t)s аппарат, устройство, применение)

Combinations of **Online Transactional Processing** - **OLTP** and **Online Analytical Processing** - **OLAP** using the same database instance application approaches are supported by SAP HANA.

**Enhancements in SAP NetWeaver 7.4 (**available since May 2013**)**

**Open SQL** is a DB abstraction layer that defines a common semantics for all SAP-supported databases. Usage of CDS views in the FROM clause of query statements and Character-like literals in the SELECT list of query statements are provided as recent Open SQL enhancements in SAP NetWeaver Application Server ABAP 7.4.

All databases certified by SAP support the recent Open SQL enhancements (eg CDS – например, CDS).

It also facilitates the development of modern browser-based and mobile applications because of its integrated **UI development toolkit for HTML5** (SAP’s adaptation of the HTML5 standard, known as **SAPUI5**) and SAP NetWeaver Gateway capabilities.

With columnar data, operations on single columns, such as searching or aggregations, can be implemented as loops over an array stored in contiguous memory locations. Such an operation has high spatial ('speɪʃ(ə)l пространственный) locality and can efficiently be executed in the CPU cache.

*Columnar data storage allows highly efficient compression.* If a column is sorted, often there are repeated adjacent values. SAP HANA employs highly efficient compression methods - run-length encoding, cluster coding, dictionary coding. *With* ***dictionary encoding****, columns are stored as sequences of bit-coded integers - each column uses an array of integer values that represent the positions of the actual values in the dictionar.*

*Columnar storage, in many cases, eliminates the need for additional index structures. Storing data in columns is functionally similar to having a built-in index for each column.* The column scanning speed of the in-memory column store and the compression mechanisms – especially dictionary compression – allow read operations with very high performance. In many cases, it is not required to have additional indexes.

*When SELECT \* is needed, go for Row (store).* If we do SELECT \* in Col Store table, then traversing through all the column store individual tiny tables would be exponentially expensive.

**SAP HANA Transport Container (HTC)**

The different ABAP and HANA development entities have to be transported through the system landscape; typically from the development system to the testing, quality system, and then to the productive system. Here comes the SAP **HANA Transport Container** - **HTC**.

*HTC is an ABAP development object which is required to integrate HANA repository content into the standard* ***Change and Transport System*** *-* ***CTS****.* It ensures an efficient delivery process of applications built out of ABAP (method) and HANA content (AMDP) or simply, ABAP for SAP HANA applications between SAP systems by means of the proven ABAP transport mechanism.

**Open and Native SQL**

***Open SQL*** *allows us to access database tables declared in the ABAP Dictionary regardless of the database platform that the R/3 System is using****.***

**Native SQL** allows us to use database-specific SQL statements in an ABAP program. This means that *we can use database tables that are not administered by the ABAP Dictionary*, and therefore integrate data that is not part of the R/3 System.

Please note, all ABAP custom code would not show drastic performance improvement automatically. In order to take maximum advantage of SAP HANA, our custom code should be in compliance with the enhanced SQL performance guidelines*.*

**ABAP Test Cockpit - ATC**

***ATC*** *can be used to check the ABAP coding*for potential functional regressions/issues and correct them(if necessary) before migrating to SAP HANA.

***ADBC*** *-* ***ABAP Database Connectivity*** *interface check in the Code Inspector should be done before migrating to SAP HANA to avoid functional gaps.*

***Runtime Check Monitor - SRTCM*** *can be used to get additional runtime information for potential functional regressions check.*

***PERFORMANCE\_DB*** *check can be used to check performance optimization potential before migrating to HANA.*

*The* ***SQL Monitor - SQLM*** *can be used to capture the*SQL profile of the ABAP system*.* SQL Monitor data can be exchanged between two systems by creating a snapshot of the SQL Monitor data, exporting it to the file system, and then importing it to the target system.

***SQL Performance Tuning Worklist - SWLT*** *allows correlating the results of an ABAP source code analysis with SQL runtime data.*

**Code to Database Paradigm Shift**

We have to consider the fact that *SAP HANA and AS ABAP use different type systems when we follow the Code-to-Data paradigm using Native SQL.*

**Core Data Services - CDS**

***CDS*** *is a****mechanism to push down logic to****database.*

CDS, are a higher-order SQL that relieves application developers from low-level SQL coding for adding referential navigation by generating the required code automatically, and also forms the basis for unified data models in the SAP HANA context. *The intention is for SAP HANA to be able to consume various data sources on the same semantic level regardless of whether they are delivered (поставляются) an ABAP program or SAP BusinessObjects model.*

*CDS view can be consumed by using the Data Preview in ABAP Development Tools for SAP NetWeaver and using it as a data source in the* ***FROM*** *clause of an Open SQL query.* Conditional expressions like **COALESCE** functions and **CASE** statement in the projection list can be used in CDS views. *We can use the static method* ***use\_features*** *of* ***class cl\_abap\_dbfeatures*** *to check if CDS views with scalar input parameters can be used in Open SQL queries in the system.*

**Map to Data source** option in SAP NetWeaver Gateway Service Builder (transaction **SEGW**) can be used to implement the consumption of a CDS view. The **DDL source** - **DDLS** in which the CDS view is defined is included in a transport request when we transport a CDS view. CDS views can also be extended using EXTEND VIEW statement programatically.

The enhancements included in CDS are –

* **Associations** on a conceptual level, replacing joins with simple path expressions in queries.
* **Annotations** to enrich the data models with additional (domain specific) metadata.
* **Expressions** used for calculations and queries in the data model.

**Association is also called Join on Demand** - *it will not be bound with the Joined table always. It kicks in the join only when we trigger the association on demand*.

Associations in a CDS view can be consumed in the FROM clause; in the WHERE and HAVING clauses and in the projection list. *The main purpose of associations in Core Data Services is to define relationships between entities.* Dictionary tables, CDS views and Dictionary views can be used as a data source in a CDS view. In simple words, ABAP Dictionary tables, CDS views, Dictionary views can be queriedin the Open SQL SELECT statement.

***Benefits of replacing JOIN statement with Association in CDS view*** *is that it can be*consumed using simple path expression and ON conditions for association are generated automatically and association can also be exposed themselves*.*

Annotation **AbapCatalog.sqlViewName** is mandatory for the definition of a CDS view.

**AbapCatalog.Buffering** annotation has scope in the entire CDS view in a Core Data Services. In other words, annotations are domain-specific metadata. **Symbol @** (at) is used to mask annotations in CDS views.

It is important to note that *scalar input parameters are database dependent* ⇒ if we plan to consume a CDS view using FROM clause of Open SQL query with scalar input then we need to keep in mind that the query cannot be executed on all SAP-certified databases for the database dependent scalar inputs.

**ABAP Managed Database Procedure - AMDP**

AMDP can be considered as a function stored and executed in the database. The implementation language varies from one database system to another. In SAP HANA it is SQL.

In order to implement an ABAP Managed Database Procedure method we need to implement (mandatory) the class interface **IF\_AMDP\_MARKER\_HDB**. ABAP language elements LANGUAGE **db\_lang** (db\_lang = SQLSCRIPT…), FOR db (db = HDB…) and BY DATABASE PROCEDURE are mandatory for the implementation of a method as an ABAP Managed Database Procedure.

*We can consume an ABAP Managed Database Procedure in your ABAP coding by calling the corresponding ABAP class method.*

**AMDP has some limitations -**

* Methods with returning parameters cannot be implemented as AMDPs.
* Method parameters have to be tables or scalar types.
* Method parameters have to be passed as values.

**ABAP Database Connectivity - ADBC**

*ADBC is an***API for the Native SQL interface***of the AS ABAP that is based on ABAP Objects.*

While the statements of Native SQL offer exclusively static access to the Native SQL interface, *ADBC makes an object orientated and dynamic access possible*.

ADBC API uses **CL\_SQL\_STATEMENT** and **CL\_SQL\_RESULT\_SET** classes. *We need to bind a reference to the internal table as an output parameter to the CL\_SQL\_RESULT\_SET instance* and fetch the result using the next\_package method of class CL\_SQL\_RESULT\_SET to retrieve the result set into an internal table after executing a Native SQL query statement to retrieve a list of information using the corresponding ABAP Database Connectivity (ADBC) API method.

ADBC is one technique to consume HANA Models in SAP ABAP Program.

# **Core Data Services**

[Содержание](#Содержание)

There are two components of CDS Views in HANA

* **DDL SQL View** - it’s read-only classical database view which is visible in ABAP Dictionary (SE11). It cannot be edited in SE11.
* **CDS View Entity** – it’s *the DDL Source File* and the actual CDS View. It’s a Database Object which is visible in Eclipse/HANA Studio/ADT and we cannot view CDS View Entity in SE11.

Обычный запрос Open SQL

select e.id, e.name, a.zipCode from Employee e

left outer join Employee2Address e2a on e2a.employee = e.id

left outer join Address a on e2a.address = a.id and a.type=’ homeAddr’

where orgunit=4711.

Этот же запрос в ABAP Open SQL

select \* from Employee into table itempl where orgunit = 4711.

loop at it.empl.

write it\_empl-id.

write it\_empl-name.

select \* from Addresse into table it\_addrs where id = employees-id.

loop at it\_addrs.

if it\_addrs-type = 'homeaddr'.

write it\_addrs-zipcode.

endif.

endloop.

endloop.

Этот же запрос в CDS

select id, name, *homeAddress*.zipCode from Employee where orgunit=4711.

There are two types of CDS Views

* **ABAP CDS**
* **HANA CDS**

With CDS, data models are**defined and consumed on the database** rather than on the server*.*

CDS is defined using an SQL-based data definition language (DDL) that is based on standard SQL with some additional concepts, such as associations, which define the relationships between CDS views and annotations, which direct (определяют) the domain-specific use of CDS artifacts.

*CDS artifacts are stored in the DDIC and can be accessed in ABAP programs via Open SQL in the same manner as ordinary ABAP tables or views.*

**What is preferred ABAP CDS or HANA CDS if the client is in ABAP on HANA DB?**

If you use ABAP on HANA DB, you can work directly on the DB and also use *HANA CDS* there. But then the CDS objects created are not managed by the ABAP Dictionary meaning you cannot access them directly with Open SQL and they are not TYPEs in the ABAP TYPE system.

**When should we use ABAP CDS and when should we use HANA CDS?**

ABAP CDS

* If you want to access the CDS entities in ABAP as data types or in Open SQL or if you want to evaluate the CDS annotations in ABAP.
* If you do not want to access the CDS entities in ABAP, but you want to transport and upgrade them like ABAP repository objects.

HANA CDS

* If you do not want to access the CDS entities in ABAP as data TYPEs or in Open SQL. An access from ABAP is then possible using Native SQL (ADBC, AMDP) only.

**Can we consume ABAP CDS natively in HANA?**

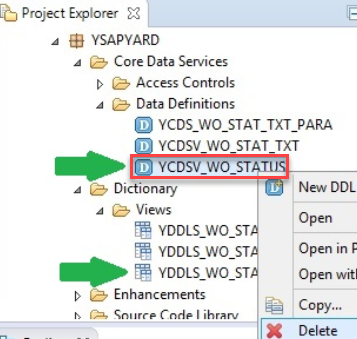
Yes we can. For each CDS view a database view (SQL view) is created in the database during activation. We can access that database view natively if we want to. *CDS table functions are managed by AMDP.* The respective database functions can also be accessed natively.

**Is it possible to access the database views (generated by having a corresponding ABAP CDS view) in HANA natively and simultaneously consider the authorization logic defined in the corresponding DCL?**

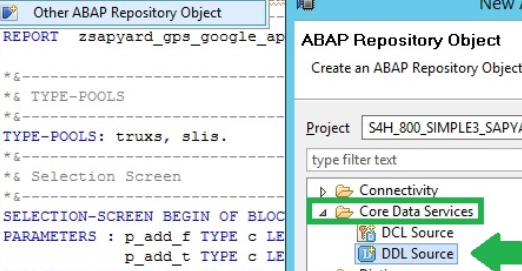
Yes. Open SQL checks the authorization implicitly but is of course translated into native SQL code doing that (который выполняется) on DB level (implicit conditions). Same for the SADL framework that checks the authorizations itself natively. *The problem is that you need to have access to the internal role representation which is not published and subject to change or you have to build a framework yourself that parses the role definition and creates the corresponding conditions.*

**What happens to DDL SQL View when CDS View (DDL Source) is deleted?**

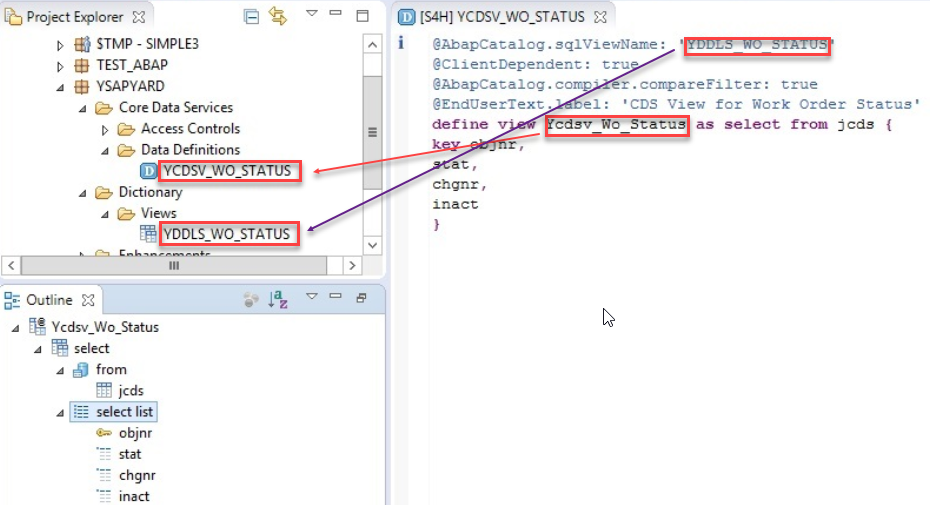
They are twins. They cannot be separated - *DDL SQL is automatically deleted when the CDS View is deleted*.



**Creating a CDS View.**

Пакет → New → 

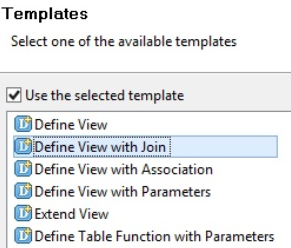
*It is a good idea to separate* ***SQL View Name*** *and actual* ***CDS View Name****.* For consistency, we name **SQL View Name with DDLS** and ***CDS View with CDSV***. You might have a different naming convention in your project.

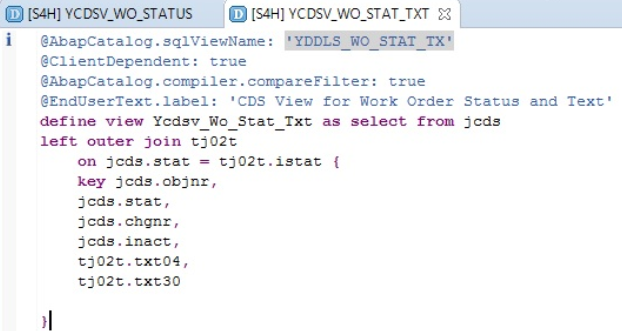


***In SE11 you can view only DDL SQL CDS View.***

*Only the CDS View entity is saved in the transport.* All change objects and transports are managed in the ABAP layer end to end.

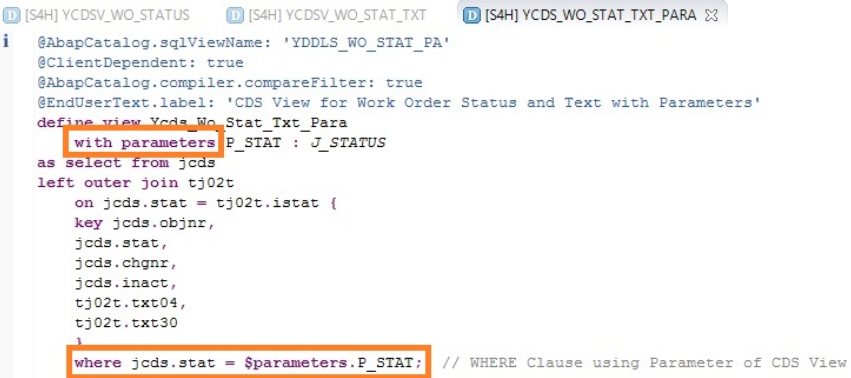
**Join in CDS View**





**Parameters in CDS View**

**$** sign needs to be provided with parameters while using it in WHERE Clause.



*We can have multiple parameters in a CDS View, separated by a comma.*

with parameters p\_stat: j\_status, p\_lang: spras

…

WHERE jcds.stat = $parameters.p\_stat and tj02t.spras = $parameters.p\_lang;

In se11 -  Data display for views with parameters is not yet supported.

**Usage of CDS View in ABAP Programs**

SELECT \* FROM ycds\_wo\_stat\_txt\_para( p\_stat = @p\_status ) INTO TABLE @i\_wo\_status.

You would notice below that “@” symbol is used for escaping of host variables**. They help to identify ABAP work areas/variables/constants in Open SQL statement**. Literals need not be escaped using “@”. If we decide to escape one host variable, all host variables should be escaped.

Also, *we can select from both DDL SQL View and CDS View* ⇒ we need to declare the internal tables/work areas according to the View you intend to use. Although **DDL SQL View** and **CDS View** are mirror images still you cannot use the TYPE statement interchangeably in the program.

**SE11 *Data Dictionary DDL SQL View* should not be normally used. Why?**

If we consume DDL SQL View in ABAP SELECT statement, then, *it will act as any other normal view/table which is created in data dictionary using SE11. We would not be taking real advantage of HANA*. We would not see the performance improvement. Theoretically, *when the DDL SQL View is used, a database connection from ABAP Layer to Database Layer is established and this process would consume some resources for database connection* (even though your database in HANA).

**Why is it good practice to use CDS View Entity (DDL Source) while using ABAP SELECT statement?**

CDS View Entity (DDL Source) is a database object which is known to ABAP Layer and does not exist in data dictionary (SE11). We can execute an SQL without creating a database connection between ABAP Layer and Database. Only results will be transferred back to ABAP layer. This will save resources for creating a database connection from ABAP Layer to Database Layer.

Example of the ABAP program to show usage of CDS View with Parameter.

REPORT YCDS\_WO\_STATUS\_REPORT.

DATA: i\_wo\_status TYPE STANDARD TABLE OF ycds\_wo\_stat\_txt\_para.

SELECTION-SCREEN BEGIN OF BLOCK a01 WITH FRAME TITLE text-001.

PARAMETERS : p\_status TYPE j\_status.

SELECTION-SCREEN END OF BLOCK a01.

START-OF-SELECTION.

PERFORM sub\_get\_data\_from\_cds.

END-OF-SELECTION.

PERFORM sub\_display\_data.

FORM sub\_get\_data\_from\_cds.

SELECT \* FROM ycds\_wo\_stat\_txt\_para( p\_stat = @p\_status ) INTO TABLE @i\_wo\_status.

ENDFORM.

FORM sub\_display\_data.

DATA:

lv\_status\_rel TYPE j\_status VALUE 'I0002', " Release Status

lr\_functions TYPE REF TO cl\_salv\_functions,

lr\_alv TYPE REF TO cl\_salv\_table,

lr\_display TYPE REF TO cl\_salv\_display\_settings,

lv\_salv\_msg TYPE REF TO cx\_salv\_msg.

IF i\_wo\_status IS NOT INITIAL.

TRY.

cl\_salv\_table=>factory( IMPORTING r\_salv\_table = lr\_alv CHANGING t\_table = i\_wo\_status ).

CATCH cx\_salv\_msg INTO lv\_salv\_msg.

MESSAGE lv\_salv\_msg TYPE 'E'.

ENDTRY.

lr\_functions = lr\_alv->get\_functions( ).

lr\_functions->set\_all( abap\_true ).

lr\_display = lr\_alv->get\_display\_settings( ).

lr\_display->set\_striped\_pattern( cl\_salv\_display\_settings=>true ).

lr\_display->set\_list\_header( text-001 ).

lr\_alv->display( ).

ELSE.

MESSAGE 'No data found' TYPE 'I'.

LEAVE LIST-PROCESSING.

ENDIF.

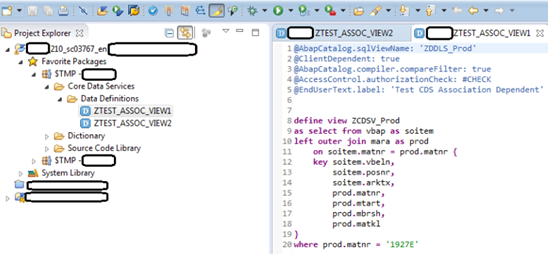
ENDFORM.

***SEGW*** *is the t-code to create OData Projects and eventually publish an OData Service.*

You can create your OData Projects without going to SEGW transaction.

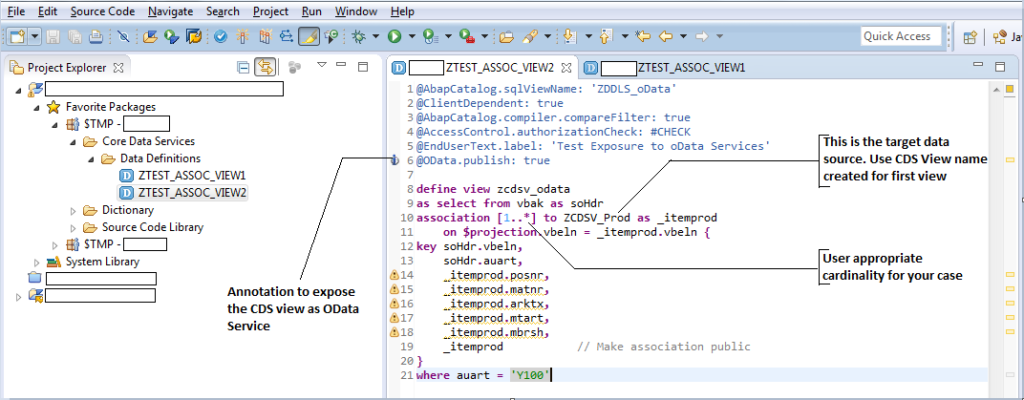
**Step 1**

*Create a view with a left outer join* between tables VBAP (soitem) and MARA (prod).



**Step 2**

*Create a second view with*[***Association***](https://sapyard.com/abap-for-sap-hana-points-to-remember/)***.*** You create an association *to* ***conceptually join or associate one data source to a target data source on a condition provided***. If data sources can be envisaged as Entities of OData service then associations are joining two entities conceptually.



Take special note of the Annotation at the 6th line: **@OData.publish: true.** This is the magic spell for our article today.

**Step – III**

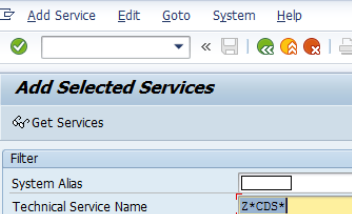
Now our view is ready. With the DDL view we should be able to see data from Header table VBAK, Item table VBAP and Product table MARA.

**Step – IV**

Note, once you activate the view you will be able to see an icon beside the annotation (6th line) “**@OData.Publish: true**” which reads that you need to ***register the service through /IWFND/MAINT\_SERVICE***.

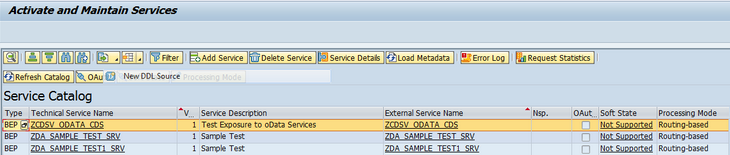
**Step – V**

Now, as instructed go to transaction /IWFND/MAINT\_SERVICE in the gateway system to **register the service** created through CDS.



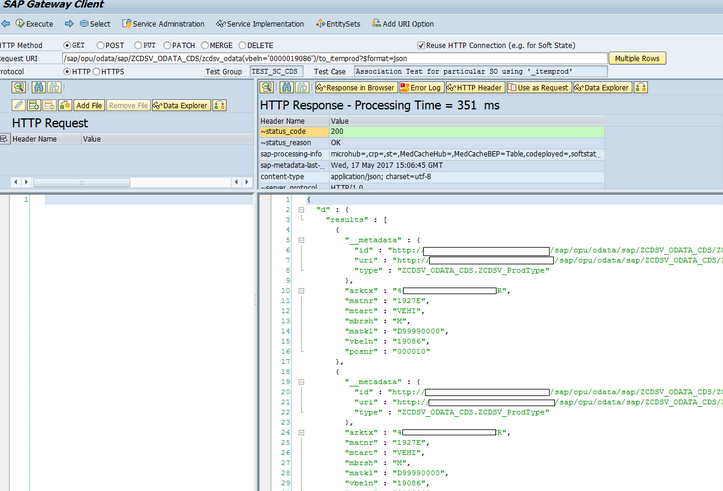
**Step – VI**

Once the service is found, click on the service to register and save it in the appropriate package. ***Note we have not used SEGW to create any service.*** This service got ***automatically generated due to OData Annotation maintained***.



**Step – VII**

Now test your service through **/IWFND/GW\_CLIENT** transaction using proper OData query. Note, for navigation unlike usual gateway, we are using **‘to\_<association name>’** in the query to navigate to the second data set. Since we created vbeln as an association condition in our ‘ZTEST\_ASSOC\_VIEW2’ the value needs to be passed through OData query for data fetching.



**Limitations**  
Please also be informed that this service can **only provide GET operation**. *No other CRUD operations can be done with this CDS view OData Exposure.*