LOGALI

Teoría

ABAP Tier 1 Cloud Ready | Tier 2 – Cloud API Enablement

SAP S/4HANA Cloud – Modelo de extensibilidad Clean Core

Contenido

1. ABAP Tier 1 Cloud Ready | Tier 2 – Cloud API Enablement

1.1. Conceptos y Objetivos

SAP Clean Core es una estrategia diseñada para mantener los sistemas SAP limpios, incorporando actualizaciones de los últimos procesos de negocio. Este enfoque es crucial para las instancias de SAP S/4HANA Cloud Public Instance y las instancias de SAP Business Technology Platform (BTP), en las que solo es posible trabajar con la versión del lenguaje ABAP for Cloud Development.

Arquitectura y Extensiones:

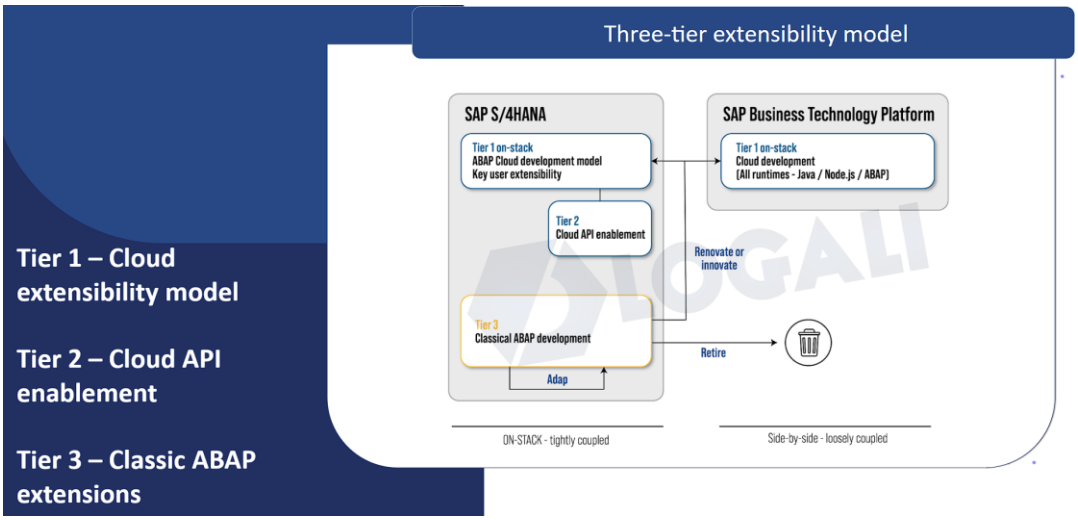
Las extensiones se dividen en dos categorías principales:

- **Extensiones Key User:** Opciones de desarrollador que se incorporan directamente al sistema.
- **Extensiones Side-by-Side:** Mantienen el sistema limpio y permiten desarrollos externos.

Niveles de ABAP:

Para las versiones de las instancias SAP S/4HANA Cloud Private Instance y on-premises, también se puede utilizar el ABAP clásico. SAP recomienda dividir el lenguaje en tres niveles (Tiers) para mantener el sistema limpio: Tier 1 (ABAP for Cloud Development), Tier 2 y Tier 3.

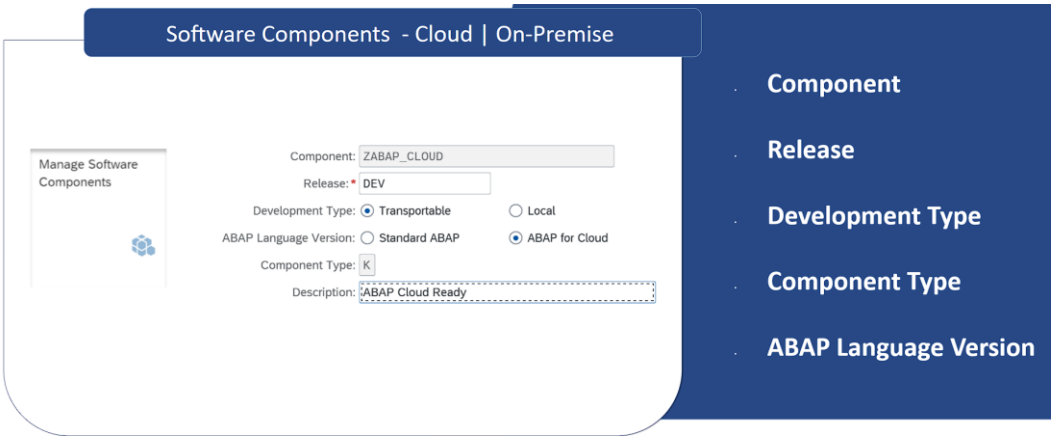
- **Tier 1:** Utilizado exclusivamente para la versión ABAP for Cloud Development.
- **Tier 2:** Enfocado en la liberación de APIs u objetos que encapsulan la lógica del Tier 3. Esto implica la creación de 'wrappers' o clases ABAP que envuelven funcionalidades no permitidas en el Tier 1, hasta que SAP libere nuevas APIs que reemplacen estos wrappers.
- **Tier 3:** Nivel donde reside la lógica principal del sistema, como BAPIs u otros objetos.



Buenas Prácticas y Herramientas

SAP sugiere prácticas específicas para asegurar el entendimiento y aplicación correcta de los Tiers:

- **Software Component:** Permite crear y vincular componentes del software para aplicar la metodología de Tiers.
- **FIORI Manage Software Components:** Utilizada en sistemas Cloud Public Edition y BTP.
- **Reportes y Programas de SAP GUI:** Utilizados en sistemas Cloud Private Edition y on-premises para crear componentes de software y aplicar la metodología de ABAP Tiers.



Aunque el ABAP clásico será retirado en el futuro, SAP recomienda comenzar a trabajar con wrappers y nuevas APIs para preparar a los desarrolladores. Este enfoque permite utilizar temporalmente la lógica del Tier 3 en el Tier 1 y facilita el mantenimiento del Clean Core.

1.2. Software Component ABAP Cloud

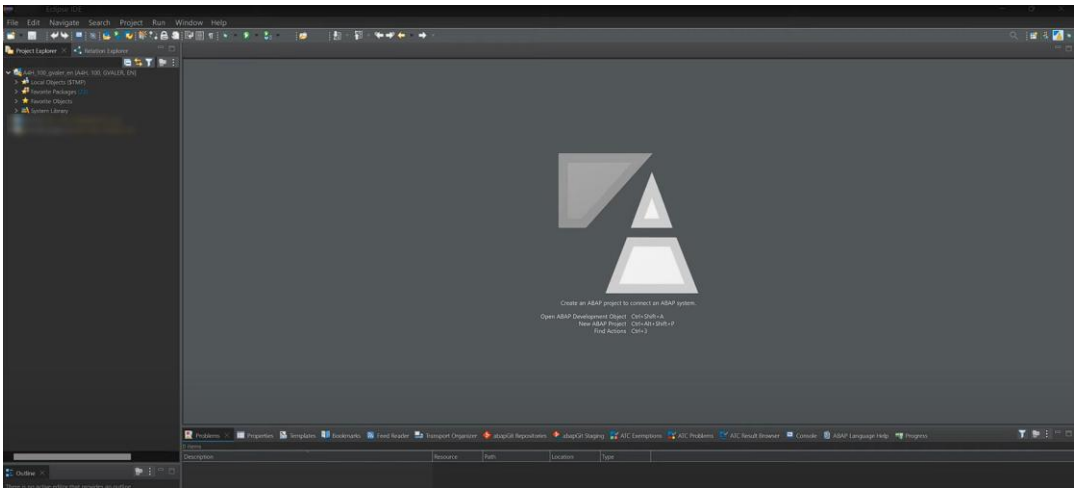
Es un componente utilizado en los sistemas SAP para agrupar objetos de desarrollo bajo un paquete de desarrollo específico. Las configuraciones realizadas en este componente determinan las condiciones y restricciones que se aplicarán a los objetos dentro del paquete de desarrollo.

En los sistemas SAP S/4HANA Cloud Public Instance y ABAP en BTP, sólo es posible utilizar la versión del lenguaje ABAP for Cloud Development. En contraste, para las versiones de SAP S/4HANA Cloud Private Edition y SAP S/4HANA on-premises, también se puede trabajar con ABAP clásico.

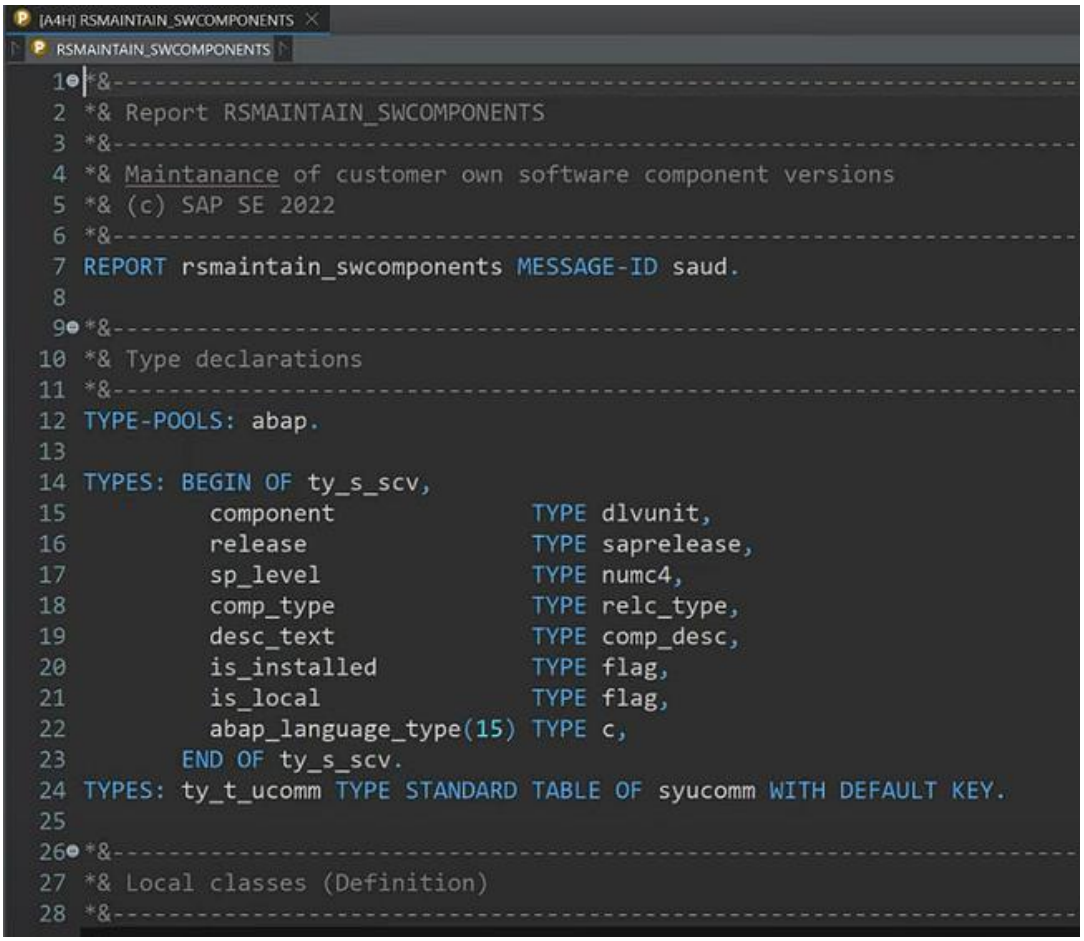
La lección se enfoca en cómo crear un Software Component que solo permita trabajar con ABAP Cloud. Este componente, al ser usado en la creación de un paquete de desarrollo, heredará las restricciones configuradas en el Software Component para establecer una línea de desarrollo clara.

Procedimiento para Crear un Software Component

Acceso al Sistema: Conectarse a un sistema que permite el uso de un programa ejecutable. Este acceso puede ser a través de Eclipse, SAP GUI o FIORI Launchpad.

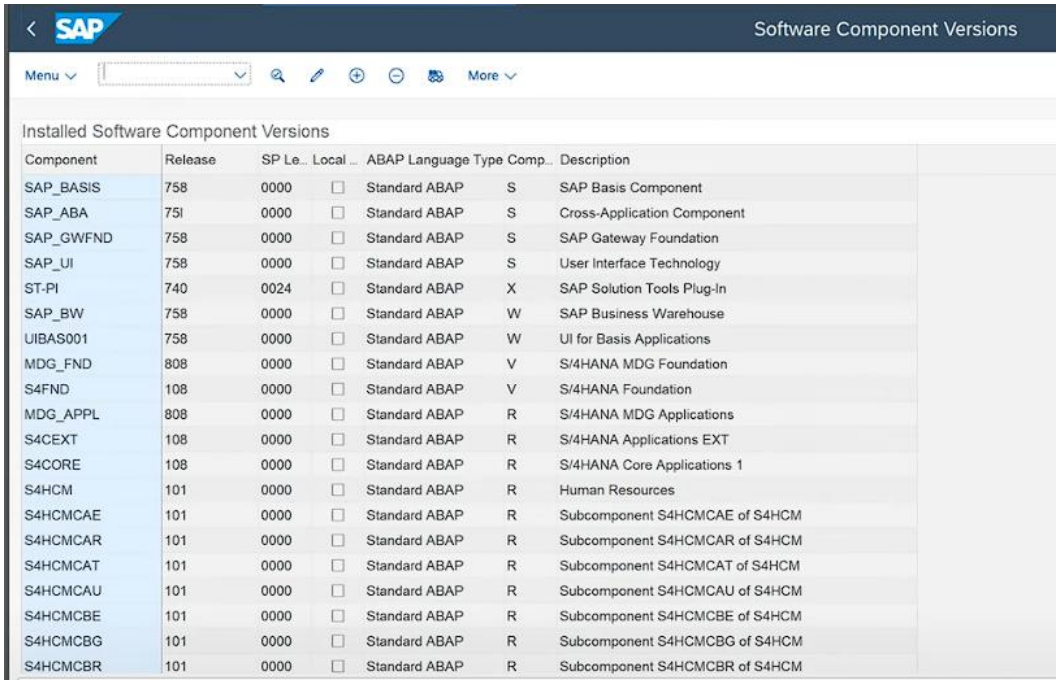


Ejecución del Programa RSMAINTAIN_SWCOMPONENTS: Este programa permite gestionar las versiones de Software Component instaladas y crear nuevas versiones.



```
[A4H] RSMAINTAIN_SWCOMPONENTS
RSMAINTAIN_SWCOMPONENTS
*&---------------------------------------------------------------------*
*& Report RSMAINTAIN_SWCOMPONENTS
*&---------------------------------------------------------------------*
*& Maintanance of customer own software component versions
*& (c) SAP SE 2022
*&---------------------------------------------------------------------*
REPORT rsmaintain_swcomponents MESSAGE-ID saud.

*&---------------------------------------------------------------------*
*& Type declarations
*&---------------------------------------------------------------------*
TYPE-POOLS: abap.

TYPES: BEGIN OF ty_s_scv,
         component             TYPE dlvunit,
         release               TYPE saprelease,
         sp_level              TYPE numc4,
         comp_type             TYPE relc_type,
         desc_text             TYPE comp_desc,
         is_installed          TYPE flag,
         is_local              TYPE flag,
         abap_language_type(15) TYPE c,
       END OF ty_s_scv.
TYPES: ty_t_ucomm TYPE STANDARD TABLE OF syucomm WITH DEFAULT KEY.

*&---------------------------------------------------------------------*
*& Local classes (Definition)
*&---------------------------------------------------------------------*
```

Al ejecutar el programa con la tecla F8, se mostrarán los diversos programas desarrollados en el sistema.



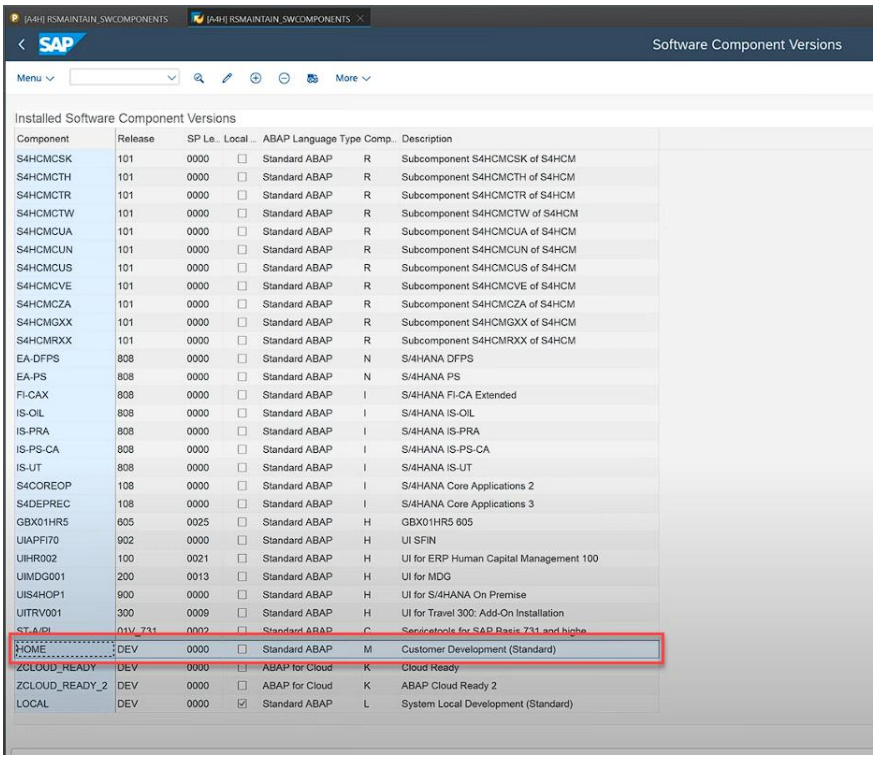
Software Component Versions

Menu More

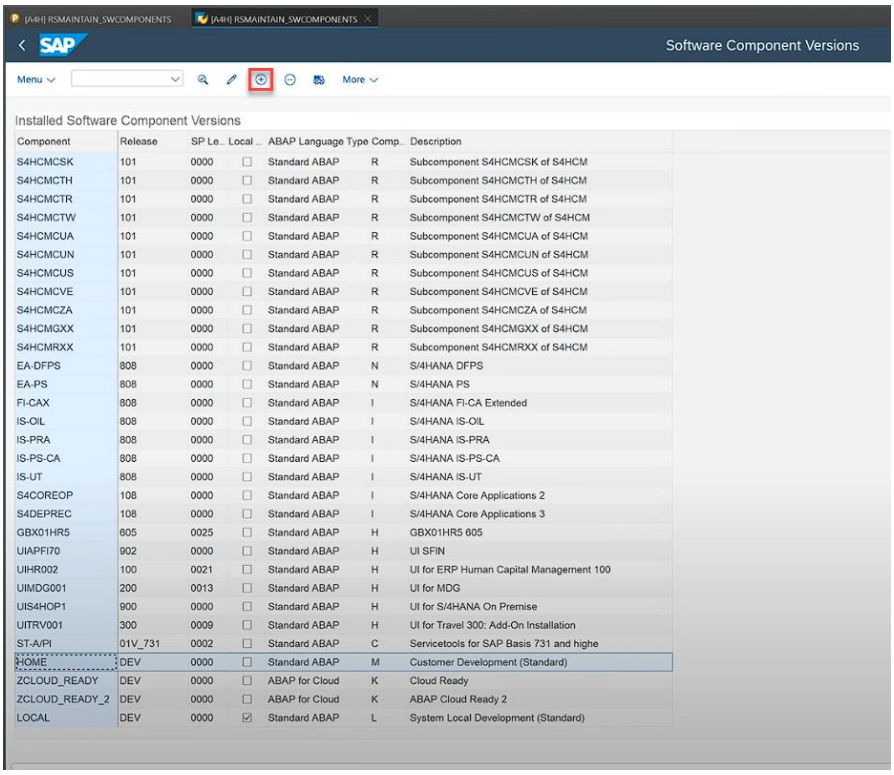
Installed Software Component Versions

Component	Release	SP Le...	Local ...	ABAP Language Type	Comp...	Description
SAP_BASIS	758	0000	☐	Standard ABAP	S	SAP Basis Component
SAP_ABA	75I	0000	☐	Standard ABAP	S	Cross-Application Component
SAP_GWFND	758	0000	☐	Standard ABAP	S	SAP Gateway Foundation
SAP_UI	758	0000	☐	Standard ABAP	S	User Interface Technology
ST-PI	740	0024	☐	Standard ABAP	X	SAP Solution Tools Plug-In
SAP_BW	758	0000	☐	Standard ABAP	W	SAP Business Warehouse
UIBAS001	758	0000	☐	Standard ABAP	W	UI for Basis Applications
MDG_FND	808	0000	☐	Standard ABAP	V	S/4HANA MDG Foundation
S4FND	108	0000	☐	Standard ABAP	V	S/4HANA Foundation
MDG_APPL	808	0000	☐	Standard ABAP	R	S/4HANA MDG Applications
S4CEXT	108	0000	☐	Standard ABAP	R	S/4HANA Applications EXT
S4CORE	108	0000	☐	Standard ABAP	R	S/4HANA Core Applications 1
S4HCM	101	0000	☐	Standard ABAP	R	Human Resources
S4HCMCAE	101	0000	☐	Standard ABAP	R	Subcomponent S4HCMCAE of S4HCM
S4HCMCAR	101	0000	☐	Standard ABAP	R	Subcomponent S4HCMCAR of S4HCM
S4HCMCAT	101	0000	☐	Standard ABAP	R	Subcomponent S4HCMCAT of S4HCM
S4HCMCAU	101	0000	☐	Standard ABAP	R	Subcomponent S4HCMCAU of S4HCM
S4HCMCBE	101	0000	☐	Standard ABAP	R	Subcomponent S4HCMCBE of S4HCM
S4HCMCBG	101	0000	☐	Standard ABAP	R	Subcomponent S4HCMCBG of S4HCM
S4HCMCBR	101	0000	☐	Standard ABAP	R	Subcomponent S4HCMCBR of S4HCM

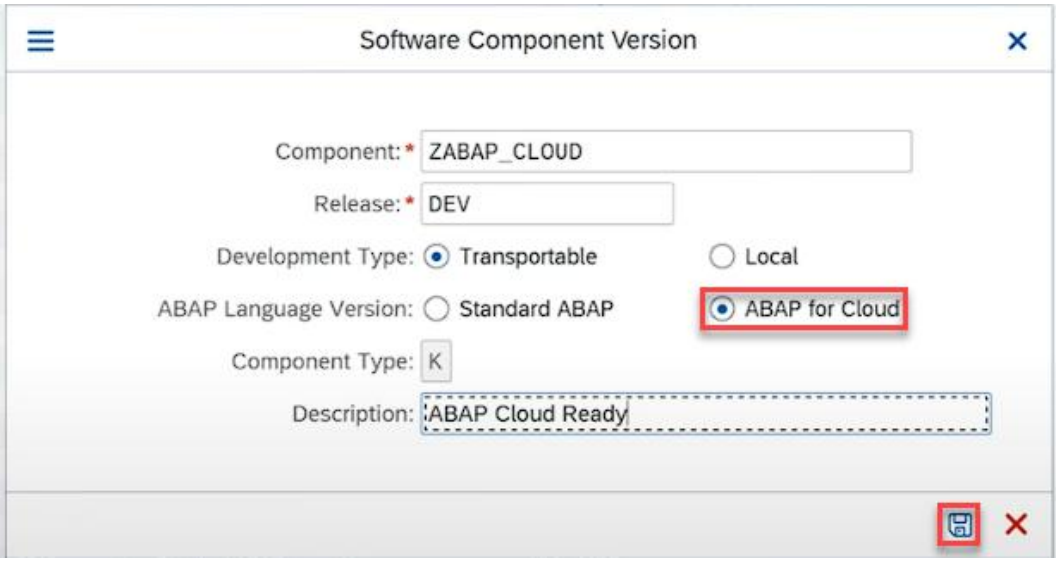
Por lo general los paquetes se asignan por defecto al Software Component “Home”.



Crear un nuevo Software Component: Para crear un nuevo Software Component. Es necesario presionar el botón con el signo “+”.

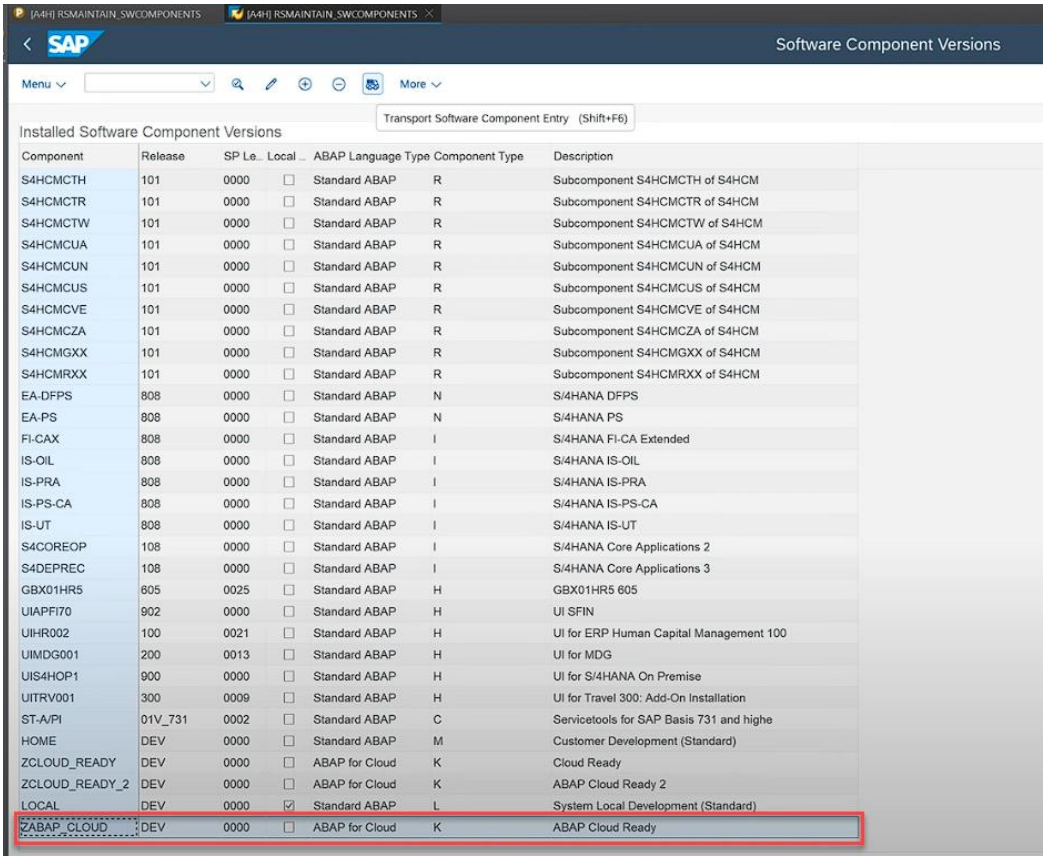


Seleccionar la versión del lenguaje para este caso ABAP for Cloud. Colocar un nombre de componente y de liberación. Además de especificar si el tipo de desarrollo es transportable o local.



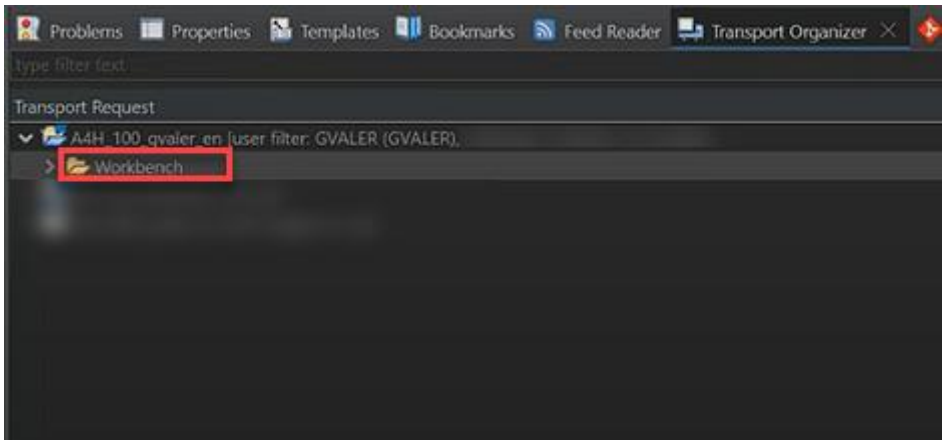
Este Software Component está configurado con el tipo de componente "K", representando la versión del lenguaje **ABAP for Cloud Development**.

Al guardar aparecerá en la lista.

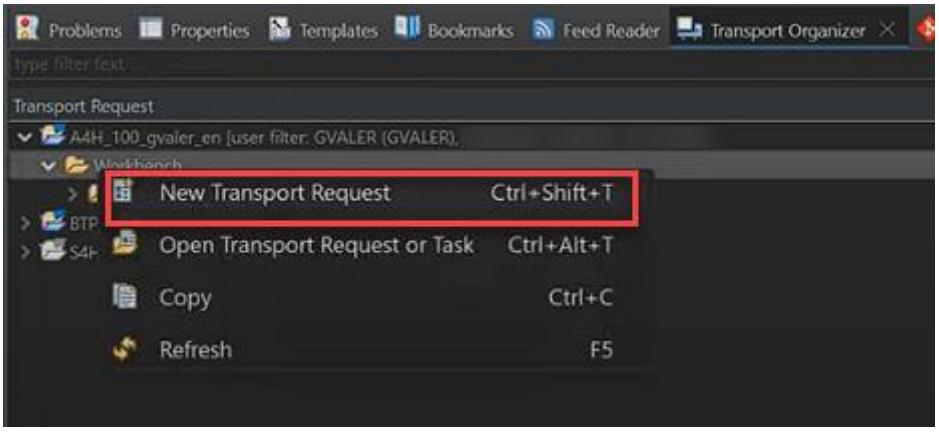


Component	Release	SP Le...	Local ...	ABAP Language Type	Component Type	Description
S4HCMCTH	101	0000	☐	Standard ABAP	R	Subcomponent S4HCMCTH of S4HCM
S4HCMCTR	101	0000	☐	Standard ABAP	R	Subcomponent S4HCMCTR of S4HCM
S4HCMCTW	101	0000	☐	Standard ABAP	R	Subcomponent S4HCMCTW of S4HCM
S4HCMCUA	101	0000	☐	Standard ABAP	R	Subcomponent S4HCMCUA of S4HCM
S4HCMCUN	101	0000	☐	Standard ABAP	R	Subcomponent S4HCMCUN of S4HCM
S4HCMCUS	101	0000	☐	Standard ABAP	R	Subcomponent S4HCMCUS of S4HCM
S4HCMCVE	101	0000	☐	Standard ABAP	R	Subcomponent S4HCMCVE of S4HCM
S4HCMCZA	101	0000	☐	Standard ABAP	R	Subcomponent S4HCMCZA of S4HCM
S4HCMGXX	101	0000	☐	Standard ABAP	R	Subcomponent S4HCMGXX of S4HCM
S4HCMRXX	101	0000	☐	Standard ABAP	R	Subcomponent S4HCMRXX of S4HCM
EA-DFPS	808	0000	☐	Standard ABAP	N	S/4HANA DFPS
EA-PS	808	0000	☐	Standard ABAP	N	S/4HANA PS
FI-CAX	808	0000	☐	Standard ABAP	I	S/4HANA FI-CA Extended
IS-OIL	808	0000	☐	Standard ABAP	I	S/4HANA IS-OIL
IS-PRA	808	0000	☐	Standard ABAP	I	S/4HANA IS-PRA
IS-PS-CA	808	0000	☐	Standard ABAP	I	S/4HANA IS-PS-CA
IS-UT	808	0000	☐	Standard ABAP	I	S/4HANA IS-UT
S4COREOP	108	0000	☐	Standard ABAP	I	S/4HANA Core Applications 2
S4DEPREC	108	0000	☐	Standard ABAP	I	S/4HANA Core Applications 3
GBX01HR5	605	0025	☐	Standard ABAP	H	GBX01HR5 605
UIAPFI70	902	0000	☐	Standard ABAP	H	UI SFIN
UIHR002	100	0021	☐	Standard ABAP	H	UI for ERP Human Capital Management 100
UIMDG001	200	0013	☐	Standard ABAP	H	UI for MDG
UIS4HOP1	900	0000	☐	Standard ABAP	H	UI for S/4HANA On Premise
UITRV001	300	0009	☐	Standard ABAP	H	UI for Travel 300: Add-On Installation
ST-A/PI	01V_731	0002	☐	Standard ABAP	C	Servicetools for SAP Basis 731 and highe
HOME	DEV	0000	☐	Standard ABAP	M	Customer Development (Standard)
ZCLOUD_READY	DEV	0000	☐	ABAP for Cloud	K	Cloud Ready
ZCLOUD_READY_2	DEV	0000	☐	ABAP for Cloud	K	ABAP Cloud Ready 2
LOCAL	DEV	0000	☑	Standard ABAP	L	System Local Development (Standard)
ZABAP_CLOUD	DEV	0000	☐	ABAP for Cloud	K	ABAP Cloud Ready

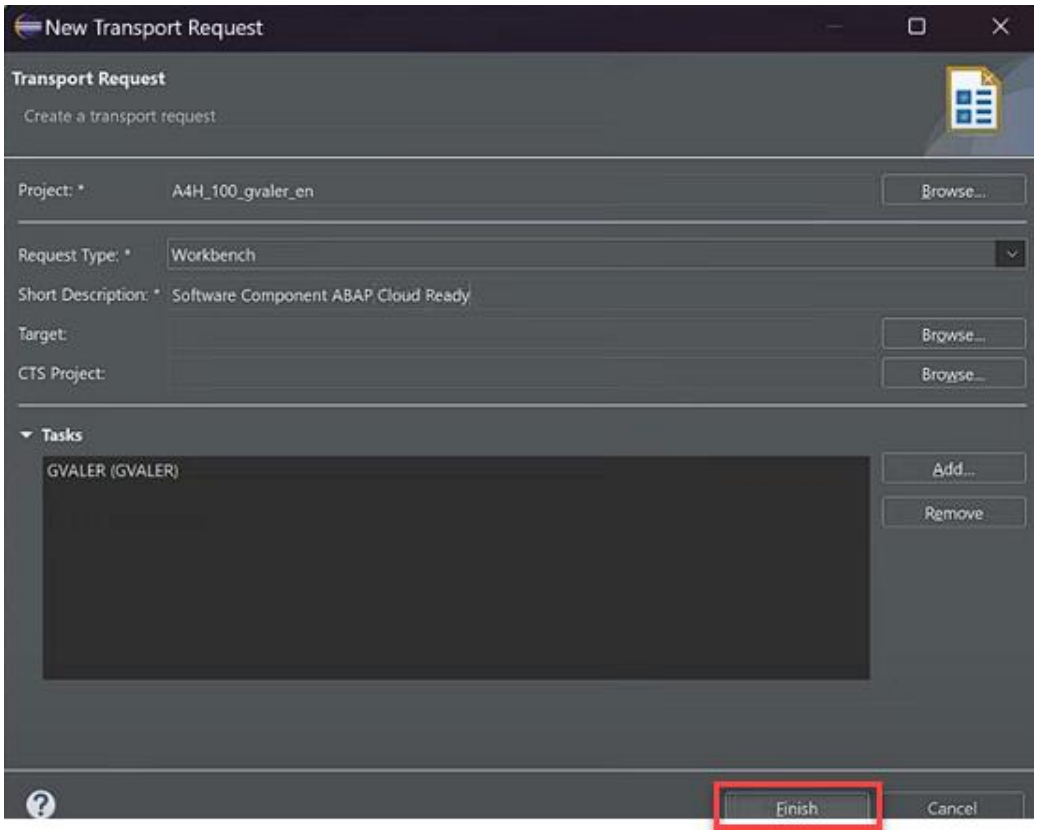
Asignación de Orden de Transporte: Luego de haber creado el Software Component es necesario asignarlo en una orden de transporte. Este proceso se puede realizar por la SE11 o por la vista Transport Organizer por Eclipse.



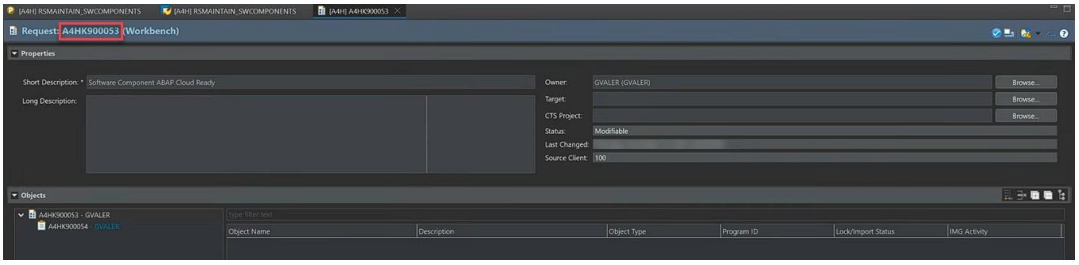
En la carpeta Workbench asociada al proyecto de desarrollo vinculado a un sistema SAP. Se debe hacer clic derecho para desplegar el menú de opciones luego, seleccionar la opción “New Transport Request”. Para crear una nueva orden de tarea.



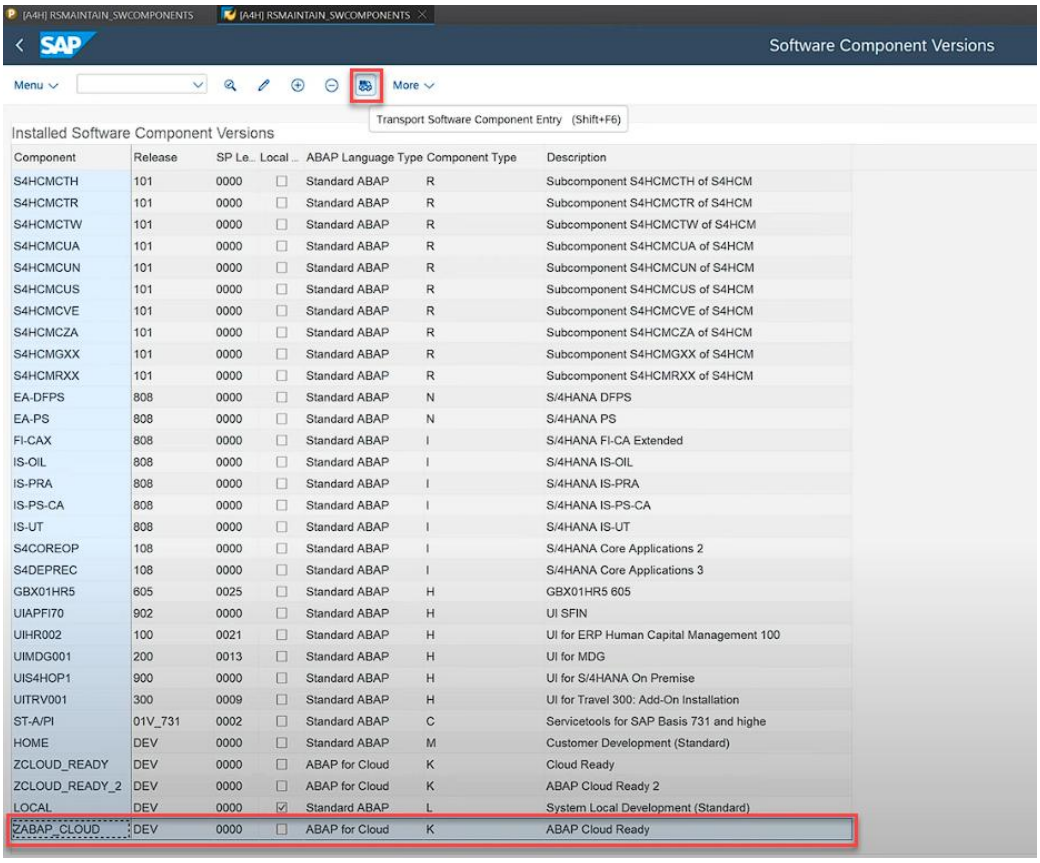
Se coloca una descripción en el campo “Request Type” y se continúa con el proceso al presionar el botón “Finish”.



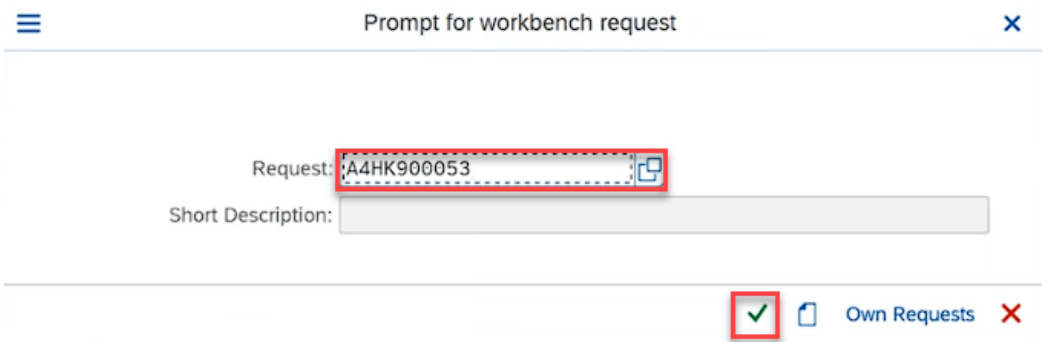
Se copia el nombre de la orden de transporte.



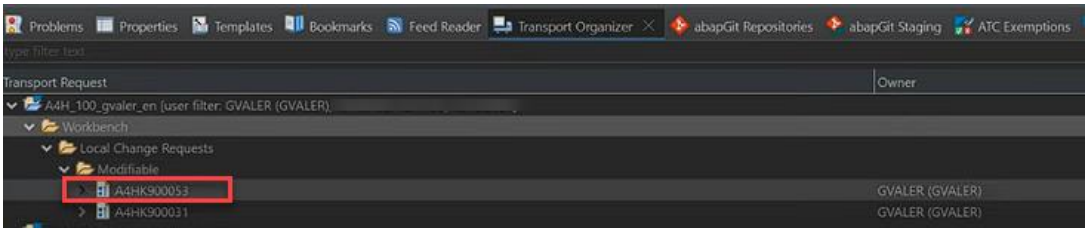
Para luego ir nuevamente a la ejecución del programa seleccionar el Software creado previamente y presionar el botón  .



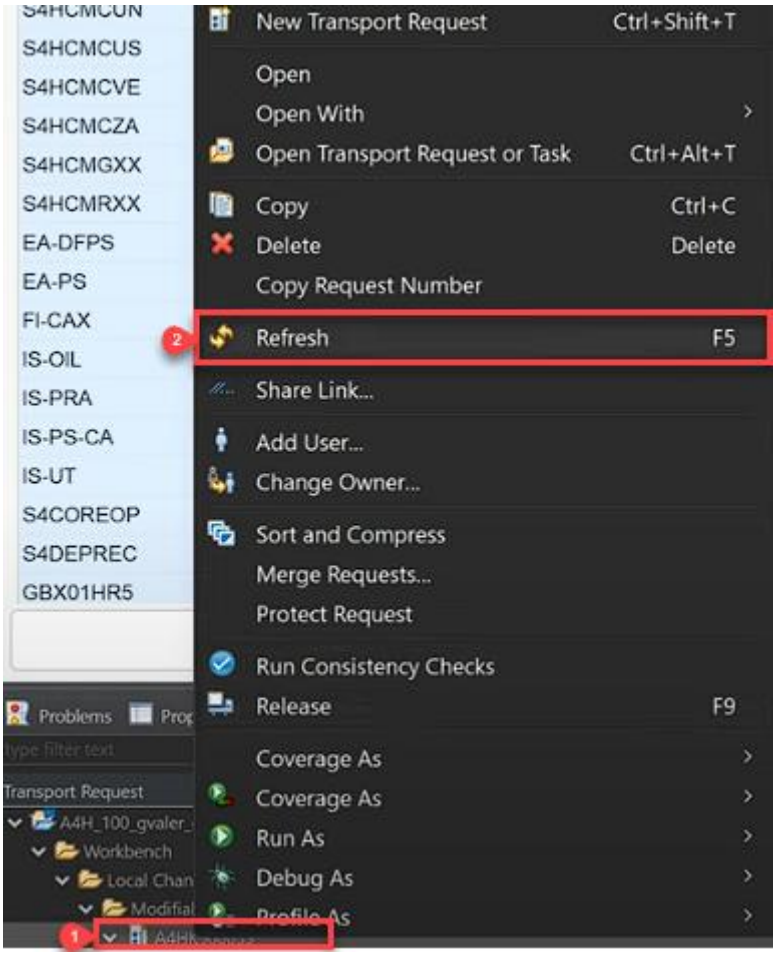
Para luego asignar el orden de transporte creada anteriormente copiando el nombre de la orden o seleccionando la por medio del match code.



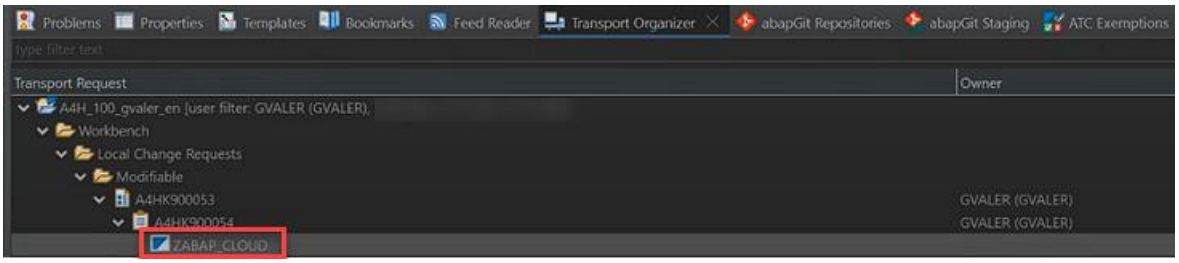
Al revisar nuevamente la vista Transport Organizer.



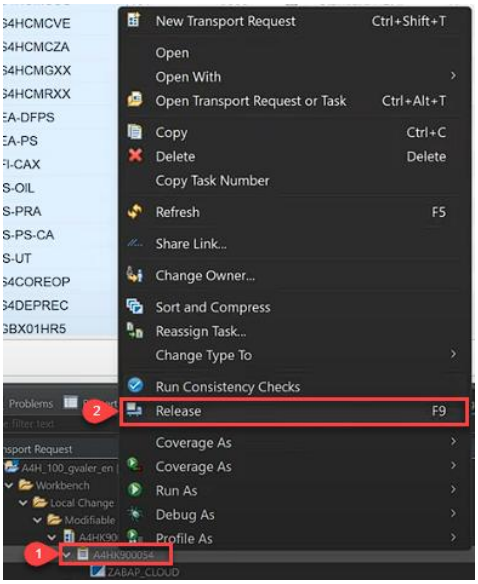
Y refrescar la orden con F5 o seleccionando la opción “Refresh”, de la lista de opciones que se despliega al hacer clic derecho en la orden.

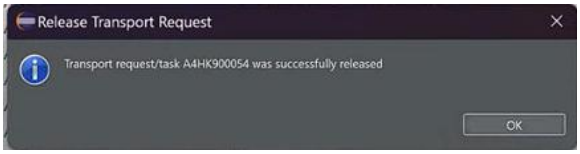


Se puede evidenciar que se ha creado una tarea con el Software Component asignado a ella.

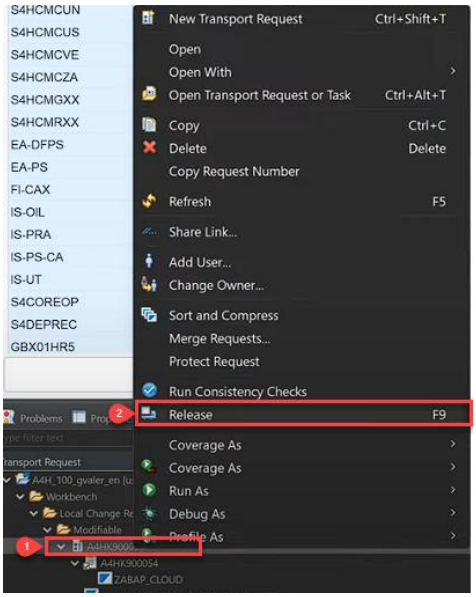


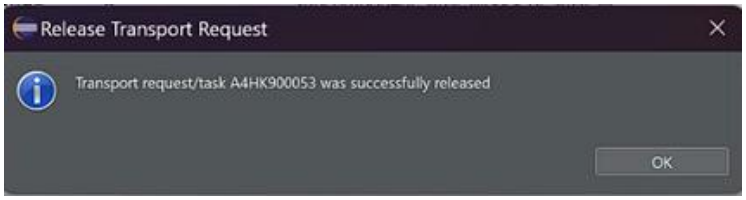
Para finalizar el proceso se debe liberar primeramente la tarea.





Y posteriormente la orden para ser transportadas al sistema objetivo.





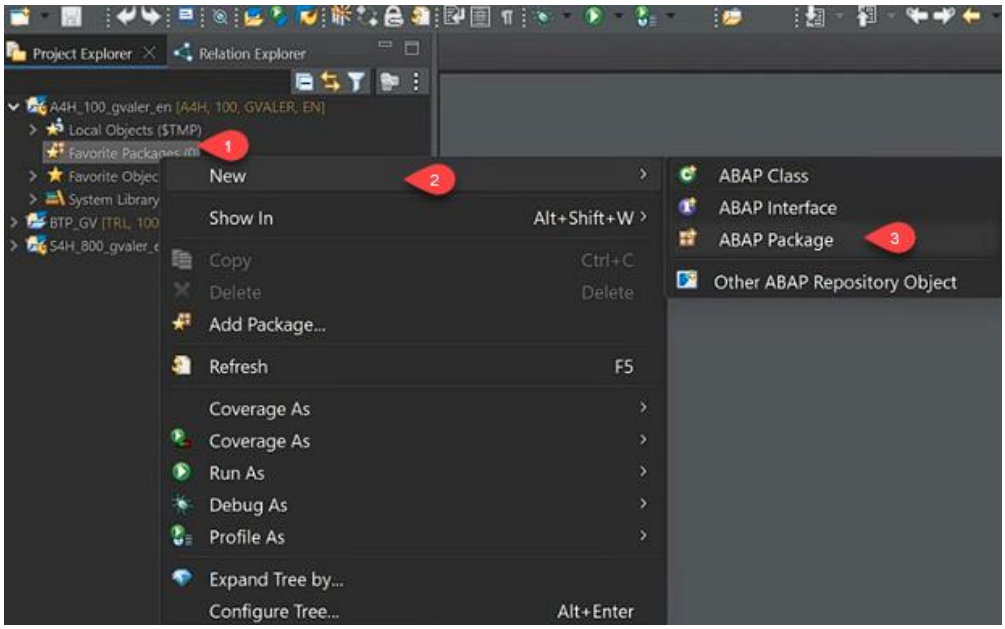
1.3. Paquete Estructura

La creación de paquetes en SAP del tipo estructura, es un proceso clave que define la estructura y organización de los objetos de desarrollo dentro de los sistemas SAP. En esta lección, se aborda la creación de un paquete de tipo estructura, el cual funcionará como el paquete padre para otros paquetes de desarrollo.

Los paquetes en SAP heredan las propiedades y configuraciones del Software Component seleccionado durante su creación. En esta lección se destaca la importancia de crear un paquete de tipo estructura, que servirá como base para los paquetes de desarrollo asignados al mismo.

Creación del Paquete en Eclipse con ADT:

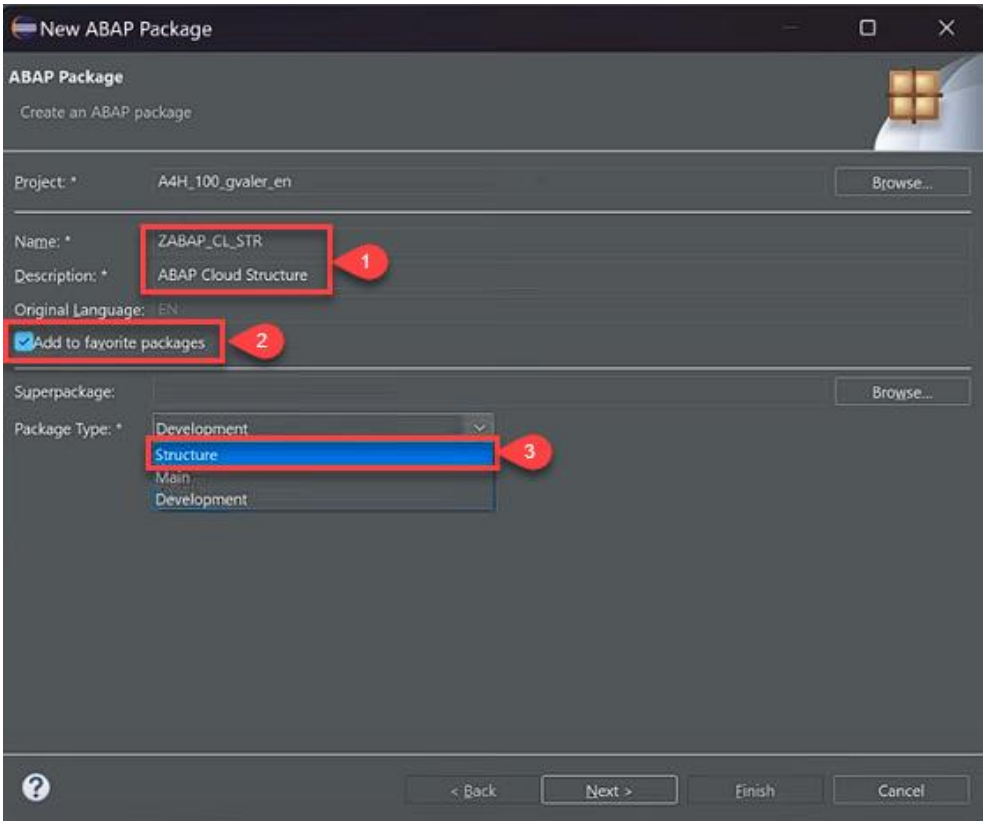
Desde un proyecto ABAP en la carpeta "Favorite Packages" en Eclipse con ADT, en una instancia ABAP Cloud Private Edition o ABAP On-premise, se inicia la creación de un nuevo paquete ABAP. Desde la carpeta "Favorite Packages" en Eclipse con ADT. Se pulsa el botón derecho del ratón sobre el nombre del paquete y se selecciona la opción "New" luego seleccionar la opción “ABAP Package” para crear un nuevo paquete ABAP.



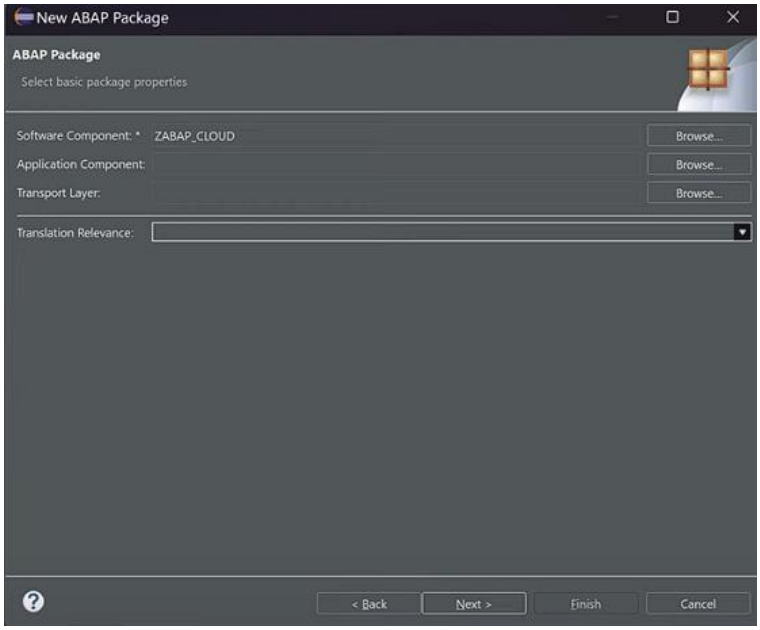
Se indica el nombre del paquete, una descripción y qué tipo de paquete debe ser “Structure”. Para el nombre del paquete se recomienda

colocar el sufijo "STR" (Structure) para denotar que es un paquete de estructura.

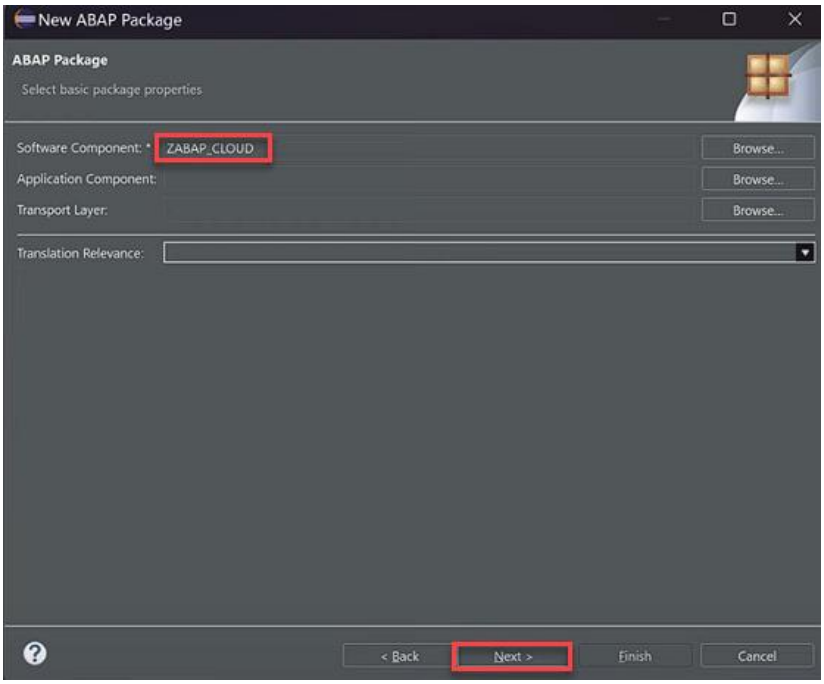
Además se puede establecer que por defecto se agregue a la carpeta de favoritos. Y continuar con el proceso presionando el botón “Next”.



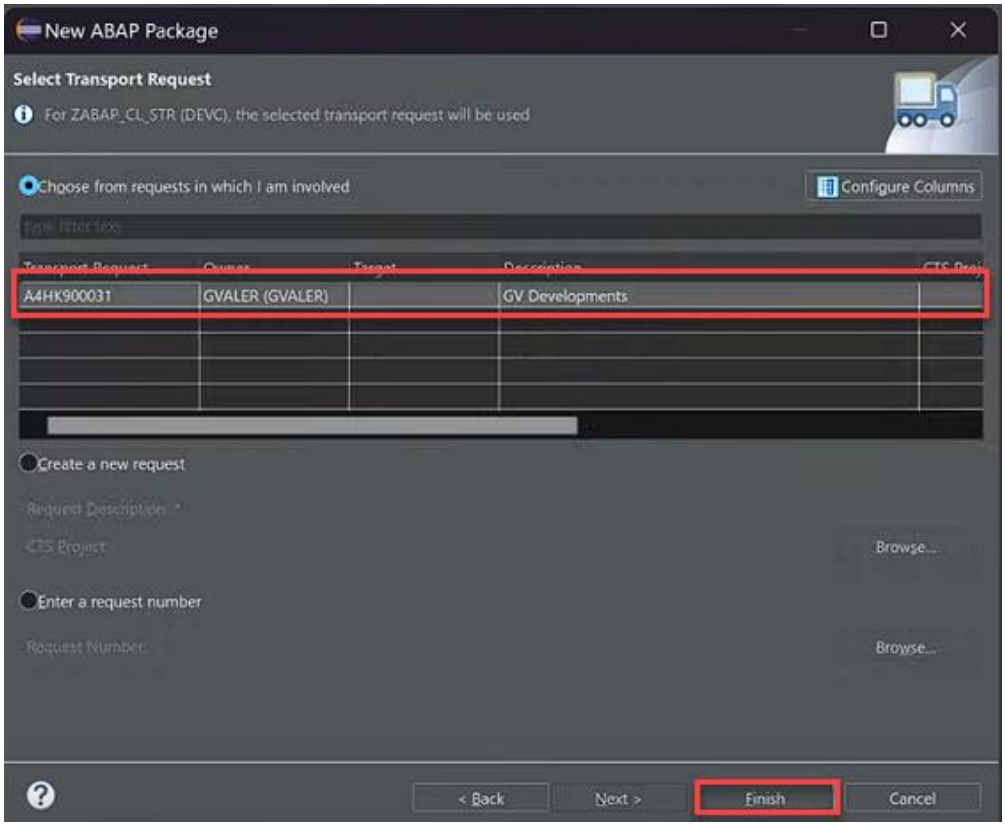
Durante el proceso de creación, es obligatorio seleccionar un Software Component.



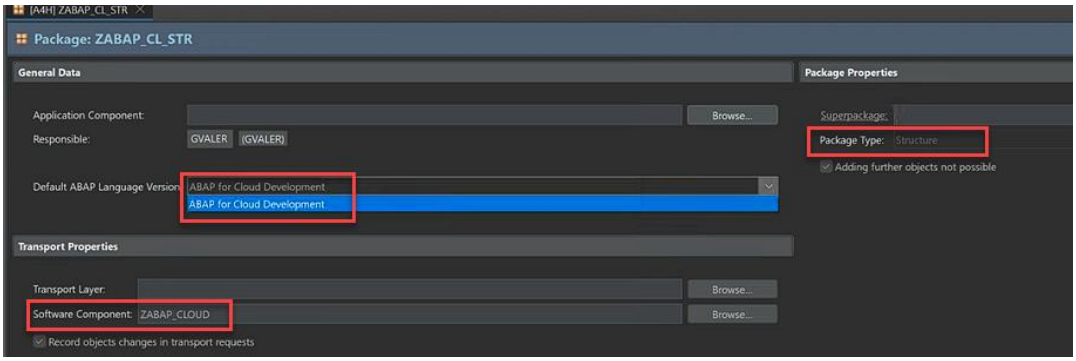
Para este caso es fundamental colocar en Software Component con una versión del lenguaje **ABAP for Cloud Development**. Y se continúa con el proceso presionando el botón “Next”.



Por último se asigna a una orden de transporte.



Se puede comprobar que el paquete recién creado tiene en la versión del lenguaje ABAP por defecto “ABAP for Cloud Development”. Su tipo de paquete es del tipo estructura.



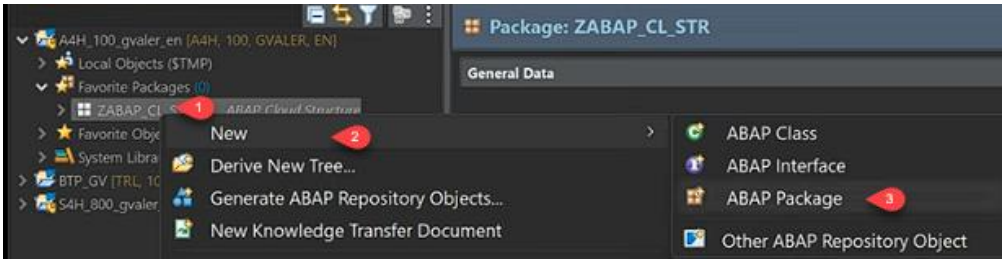
1.4. Paquete de Desarrollo – ABAP Cloud

La creación de Paquetes de Desarrollo en SAP es un proceso detallado que implica asignar configuraciones específicas heredadas desde un paquete de tipo estructura previamente creado. Esta lección se enfoca en la creación de un paquete de desarrollo que utilizará el mismo Software Component que el paquete de tipo estructura, asegurando una organización coherente y estructurada para los desarrollos en ABAP Tier 1.

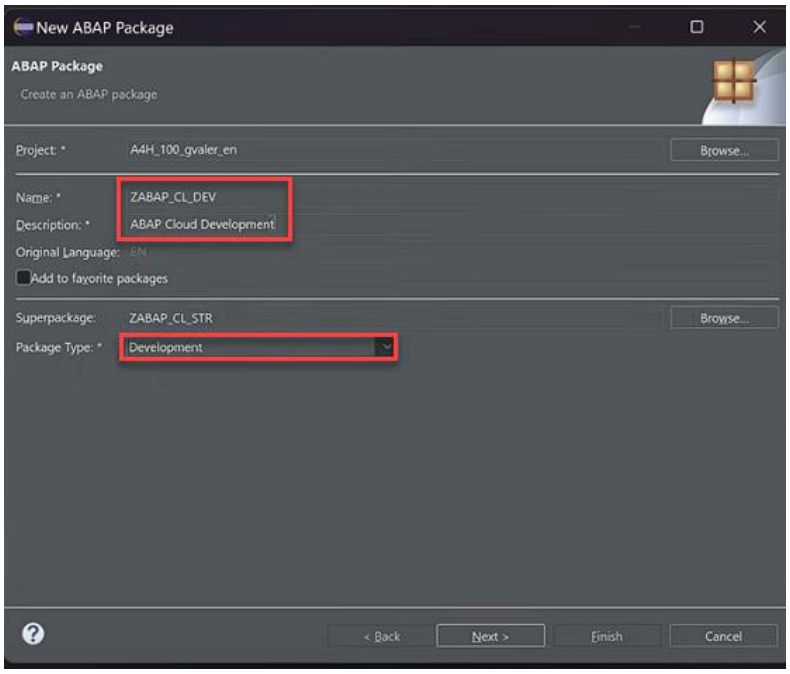
Los paquetes de desarrollo heredan las propiedades y configuraciones del Software Component asignado durante su creación. En esta lección, se resalta la importancia de crear paquetes de desarrollo que mantengan la consistencia y las restricciones del Software Component, particularmente en entornos ABAP Cloud Private Edition y on-premise.

Procedimiento para la Creación de un Paquete de Desarrollo

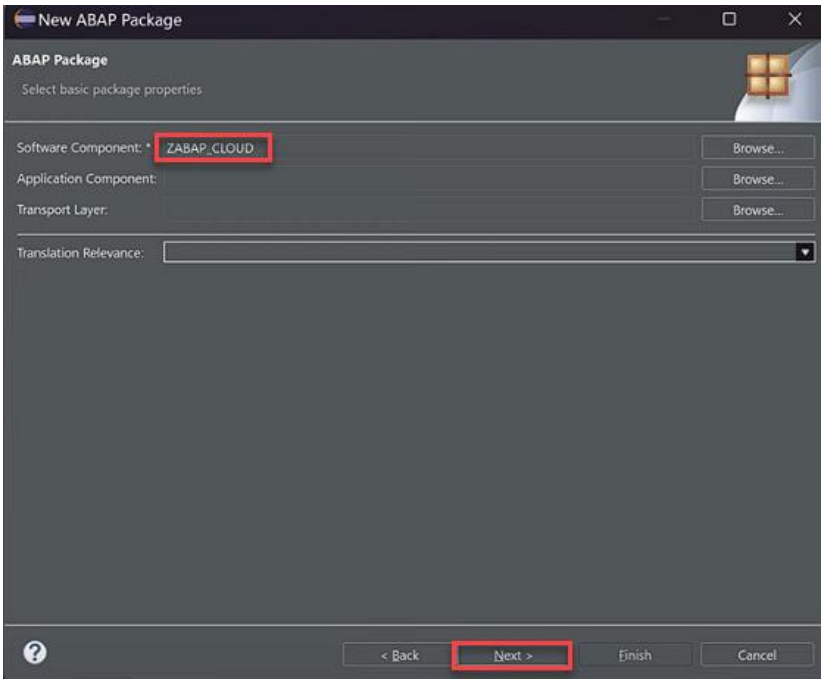
Acceso y Selección del Paquete de Estructura: Desde la carpeta "Favorite Packages" en Eclipse con ADT, se identifica el paquete de tipo estructura ya existente. Se pulsa el botón derecho del ratón sobre el nombre del paquete y se selecciona la opción "New" luego “ABAP Package” para crear un nuevo paquete ABAP.



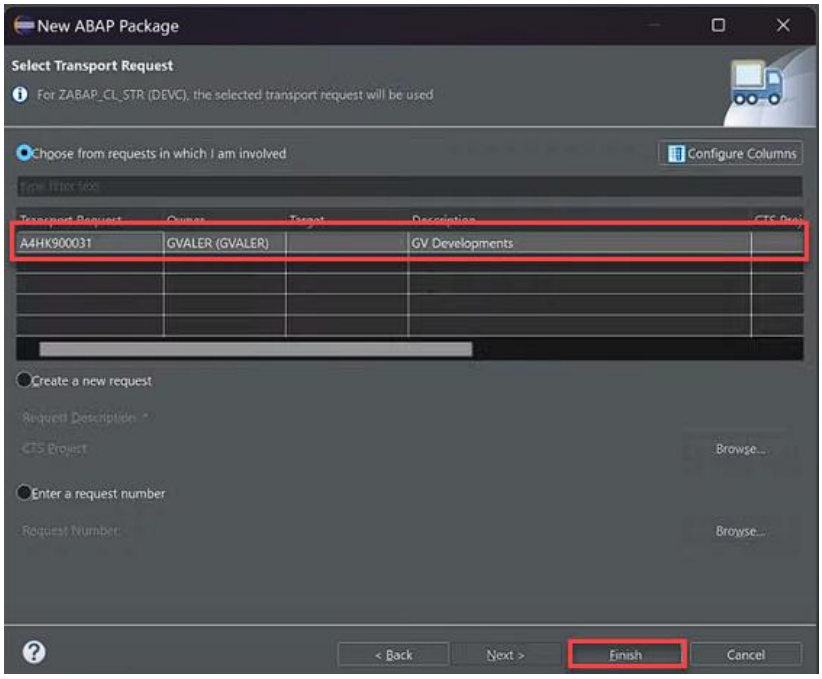
Creación del Nuevo Paquete de Desarrollo: Se coloca un nombre y una descripción. El tipo de paquete se deja como predeterminado "Development". Por defecto el paquete de estructura se agrega como subpaquete.



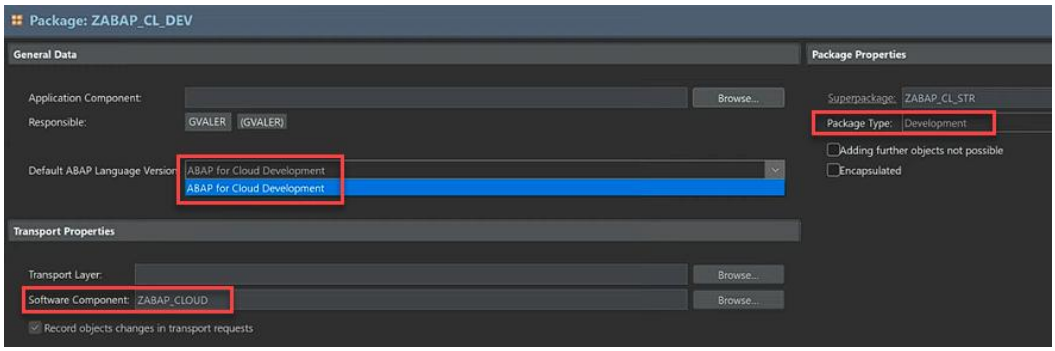
Asignación del Software Component: Se continúa con la asignación del mismo Software Component heredado del paquete de estructura, asegurando que todas las configuraciones y restricciones se mantengan. Se asigna una orden de transporte que permitirá mover los objetos de desarrollo entre diferentes ambientes.



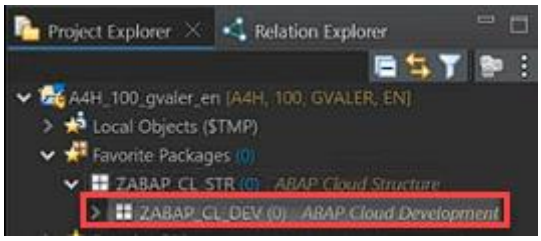
Por último se asigna la creación del paquete de desarrollo a una orden de transporte.



Se puede comprobar que el paquete recién creado tiene en la versión del lenguaje ABAP por defecto “ABAP for Cloud Development”. Su tipo de paquete es del tipo de desarrollo.



Además que el paquete aparecerá representado dentro del paquete del tipo estructura creado anteriormente.



1.5. Restricciones Tier 1

Es un proceso que asegura el cumplimiento de las mejores prácticas en los desarrollos de SAP, particularmente en entornos S/4HANA Cloud Private Instance y S/4HANA on-premises. Estas restricciones se basan en la necesidad de trabajar exclusivamente con la versión del lenguaje ABAP for Cloud Development y no con el ABAP estándar.

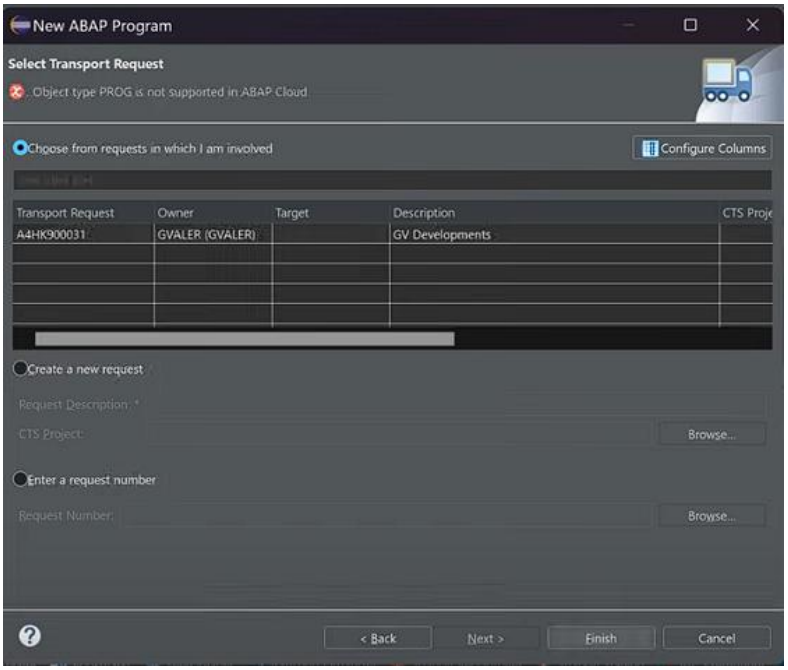
En los entornos mencionados, es posible trabajar aún con ABAP estándar, lo que permite crear o utilizar algunos artefactos de desarrollo disponibles solo en esa versión del lenguaje. Sin embargo, para mantener un entorno limpio y compatible con futuras actualizaciones, es crucial definir y respetar las restricciones para desarrollos en ABAP Tier 1.

En los paquetes que tengan configurado el Software Component la versión lenguaje **ABAP for Cloud Development.** Los desarrollos tendrán restricciones del ABAP Tier 1 para asegurar un entorno de desarrollo limpio y compatible. Esto incluye evitar el uso de artefactos y sintaxis no soportados, y trabajar dentro de las configuraciones establecidas en el Software Component.

Restricciones Específicas del ABAP Tier 1:

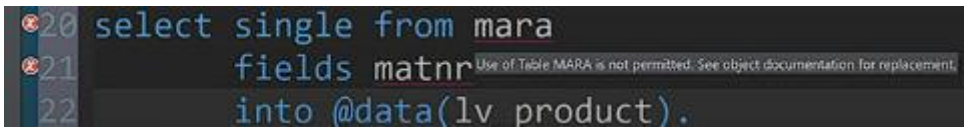
Algunas restricciones Específicas del ABAP Tier 1 tenemos:

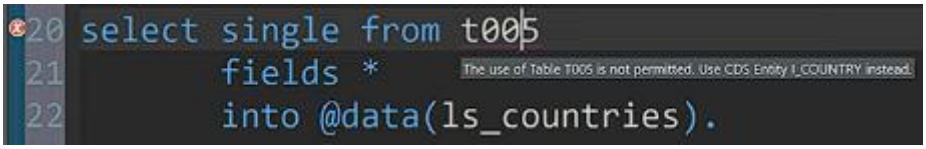
- **Programas ejecutables no permitidos:** Los programas ejecutables no están soportados en **ABAP for Cloud Development**. De hecho al intentar crearlos y asignarlos a una orden de transporte generará un error por la versión del lenguaje. En su lugar si está permitido programación orientada a objetos dentro del entorno de ABAP for Cloud Development por medio del uso de clases.



- **Tablas Estándar No Permitidas**: No es posible utilizar tablas estándar como "MARA" dentro de la versión del lenguaje **ABAP for Cloud Development**.

Esto significa que no se pueden realizar selecciones de datos directas (SELECT) desde estas tablas.





- **Objetos Liberados por SAP**: Solo se pueden utilizar objetos liberados por SAP, como las CDS (Core Data Services) que reemplazan las consultas a las tablas estándar no permitidas.

```
select single from I_Country
       fields *
       into @data(ls_countries).
```

En este caso se utiliza la CDS **I_Country**, para realizar una consulta a la tabla de sistema **T005**.

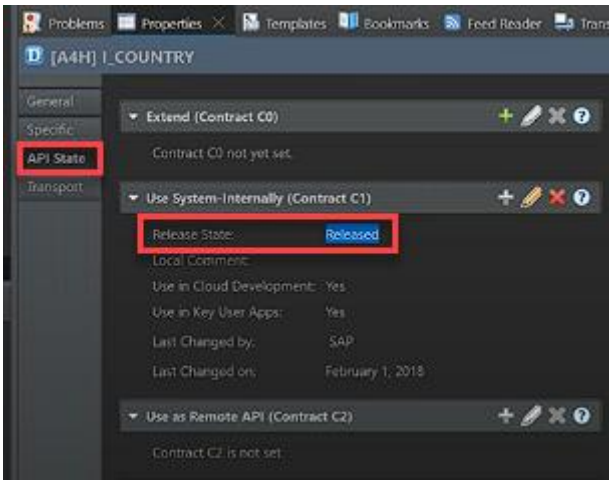
```
@ClientHandling.algorithm: #SESSION_VARIABLE
@AbapCatalog.buffering.status: #ACTIVE
@AbapCatalog.buffering.type: #FULL
@AbapCatalog.buffering.numberOfKeyFields: 1
@Metadata.ignorePropagatedAnnotations: true

define view I_Country
  as select from t005
  association [0..*] to I_CountryText as _Text on $projection.Country = _Text.Country
{
      @Search.defaultSearchElement: true
      @Search.fuzzinessThreshold: 0.8
      @ObjectModel.text.association: '_Text'
  key land1  as Country,
      intca3 as CountryThreeLetterISOCode,
      intcn3 as CountryThreeDigitISOCode,
      intca  as CountryISOCode,
      xegld  as IsEuropeanUnionMember,
```

En el caso de la tabla MARA, se utilizará la CDS I_Products para realizar consultas a esta tabla de base de datos.

```
    select from I_Product
                fields *
                into table @et_products
                up to 8 rows.
```

```
@ObjectModel.alternativeKey:[{id:'OID',element:['ProductOID']}]
@ObjectModel.sapObjectNodeType.name:'Product'
@ObjectModel.objectIdentifier.oidElement:'ProductOID'


define view I_Product
  as select from mara
  //    left outer join maw1 on mara.matnr = maw1.matnr //moved to _ProductRetail
  association [0..*] to I_ProductText               as _Text
  association [0..1] to I_ProductGroup              as _ProductGroup
  association [0..*] to I_ProductGroupText          as _ProductGroupText
  association [0..1] to I_ProductGroup_2            as _ProductGroup_2
  association [0..*] to I_ProductGroupText_2        as _ProductGroupText_2
  association [0..1] to I_ProductRetail             as _ProductRetail
  association [0..1] to I_Division                  as _Division
  association [0..*] to I_DivisionText              as _DivisionText
  association [0..1] to I_ExtProdGrp                as _ExternalProductGroup
  association [0..*] to I_ExtProdGrpText            as _ExtProdGrpText
```

Este estatus liberado puede ser comprobado en la vista “Properties” en el apartado “API State” el campo “Release State” con el estatus “Released”.



Estas restricciones son esenciales para:

- **Mantener la Compatibilidad:** Aseguran que el código desarrollado sea compatible con futuras actualizaciones de SAP.
- **Cumplir con Normativas:** Garantizan que los desarrollos sigan las mejores prácticas y normativas establecidas por SAP.
- **Optimizar el Desempeño:** Promueven el uso de objetos y métodos eficientes y soportados por la versión de lenguaje actual.

1.6. Uso de Objetos entre Software Component diferentes

Es un proceso que permite la reutilización de objetos de desarrollo, promoviendo una estructura eficiente y organizada. Esta lección se centra en la utilización de objetos creados en diferentes Software Components, explicando paso a paso cómo lograr este objetivo.

En los sistemas SAP, los objetos de desarrollo pueden estar aislados dentro de sus respectivos Software Components. Sin embargo, para una organización más eficiente y la reutilización de código, es crucial entender cómo habilitar el uso de estos objetos entre diferentes Software Components.

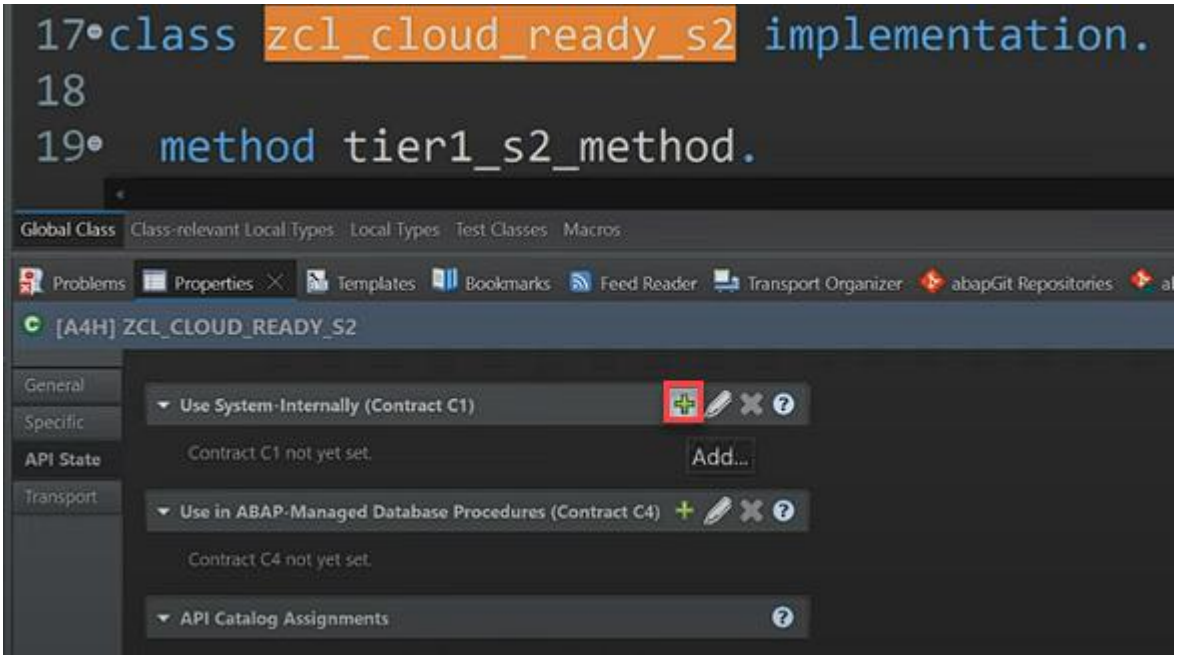
Restricción Inicial:

Al intentar utilizar un objeto de un Software Component que se encuentra en otro. Esto no es posible debido a la restricción de

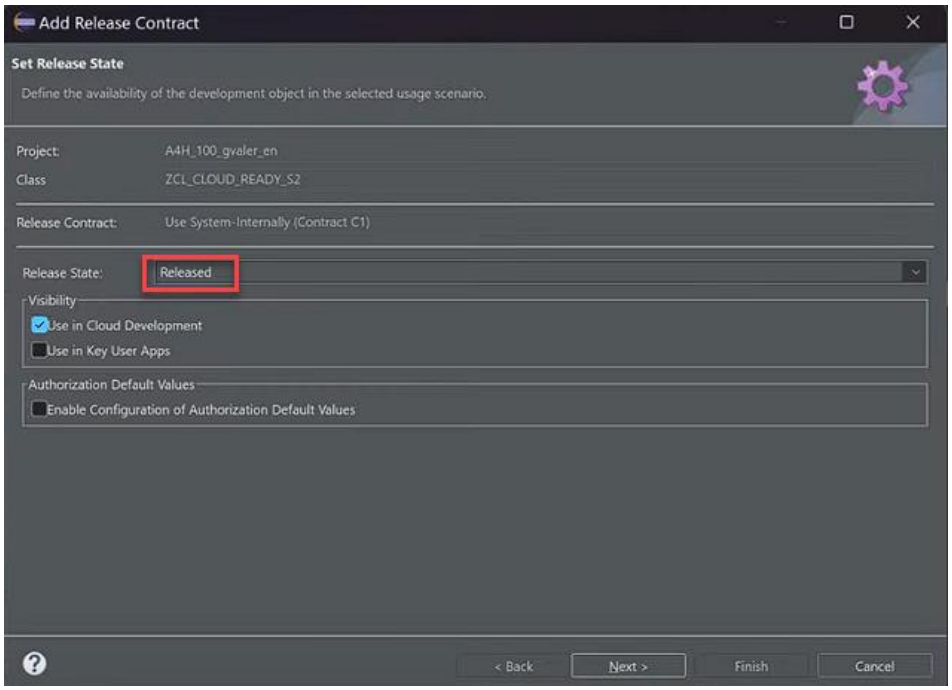
aislamiento. Cada objeto creado en un Software Component está inicialmente limitado a ser utilizado solo dentro de ese componente.

Proceso de Liberación:

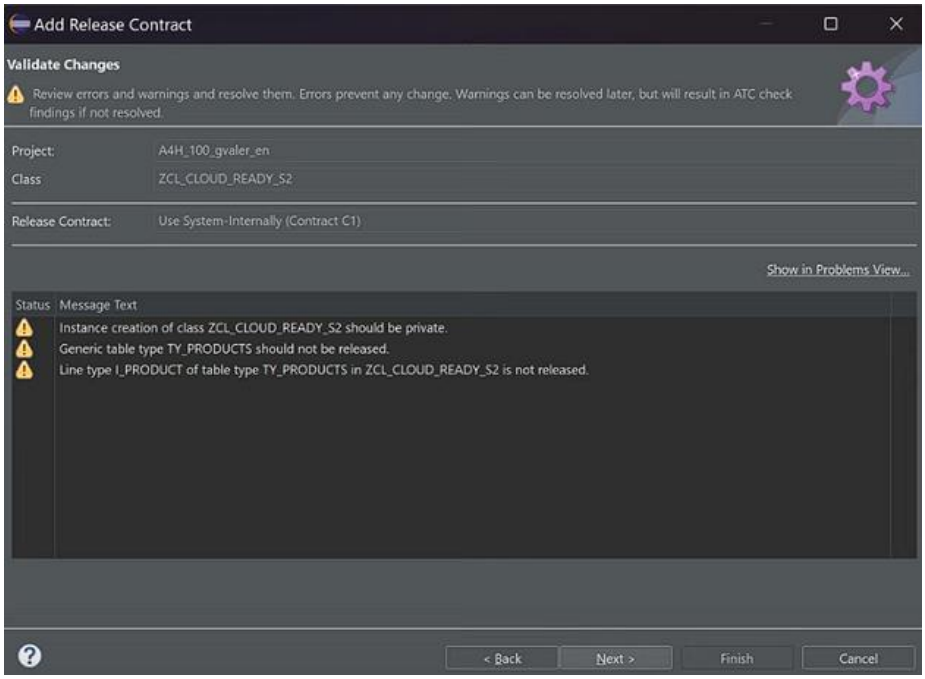
Para permitir el uso del objeto en otro Software Component, se debe liberar el objeto. Para realizar este proceso se debe acceder a las propiedades del objeto y en la pestaña **API State**, presionar el botón con el signo “+” en la sección “Use System-Internally (Contract C1)”.



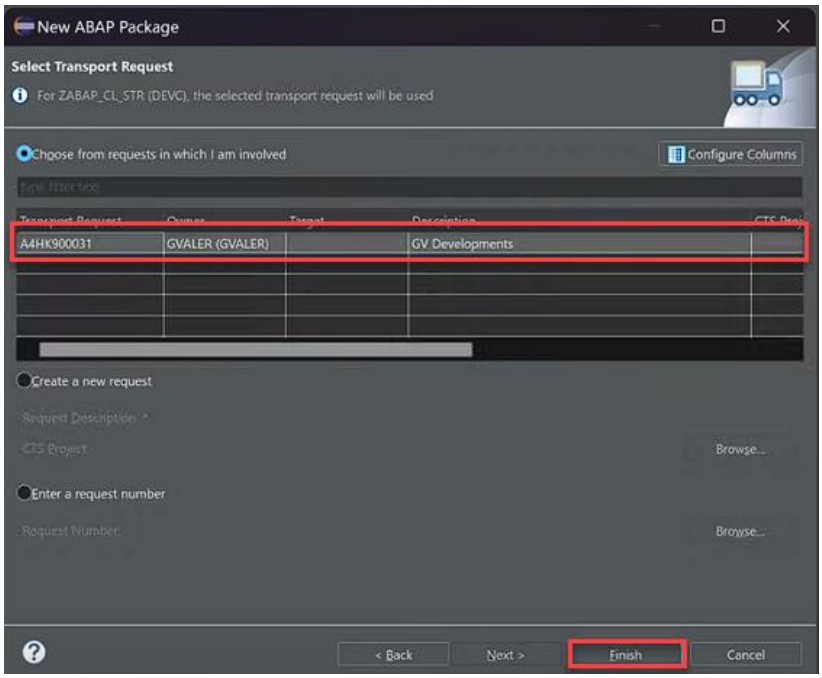
Para cambiar el contrato a un estado liberado ("Released"), y continuar el proceso presionando el botón “Next”.



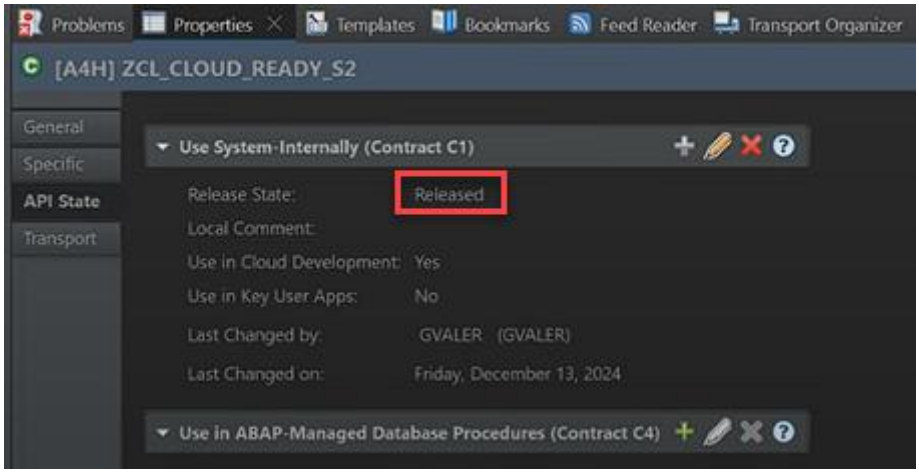
Igualmente se presiona el botón “Next”.



Se asigna el cambio a una orden de transporte.



Ahora el estatus de liberación se encontrará con el valor “Released”.



Permitiendo de esta manera utilizar el objeto en otro Software Component.

```
24    data(lo_cloud) = new zcl_cloud_ready_s2(  ).
```

Es fundamental seguir buenas prácticas para la liberación, como asegurarse de que las instancias sean privadas y que los objetos utilizados también estén liberados.

1.7. Tier 2 - Interfaz Wrapper para BAPI

En un caso empresarial que se centra en el uso de instancias de SAP S/4HANA Cloud Private Edition y SAP S/4HANA On-Premise. Debido a las limitaciones del lenguaje ABAP en estas versiones, se propone una solución que permite trabajar en el "Tier 1" utilizando ABAP for Cloud Development con APIs liberadas por SAP, y en el "Tier 2" habilitando APIs a través de wrappers. Esta técnica permite trabajar con objetos de negocio que aún residen en el clásico ABAP y que no han sido liberados como APIs en el Tier 1.

La necesidad de esta solución surge porque SAP aún no ha liberado ciertas APIs, aplicaciones RAP, o modelos con CDS (Core Data Services). Por lo tanto, es crucial que las empresas tengan la capacidad de interactuar con estos objetos de negocio a través de las BAPIs (Business Application Programming Interfaces) en los procesos de negocio.

El término API no se limita a servicios web o protocolos de comunicación como CRUD (Create, Read, Update, Delete). Una API es una interfaz que permite realizar procesos de negocio complejos a través de encapsulaciones. En el contexto del SAP, esto puede incluir CDS, objetos RAP, y más.

Recomendación de SAP:

SAP recomienda que los clientes utilicen vías como Customer Influence, la ayuda de SAP, contactos directos con SAP, y la apertura de tickets para indicar la necesidad de liberar una API específica. SAP tomará acción sobre estas solicitudes y las agregará a su cola de tareas para futuras liberaciones. Mientras tanto, se sugiere trabajar en el Tier 2 con el clásico ABAP, creando wrappers para las BAPIs.

Wrapper:

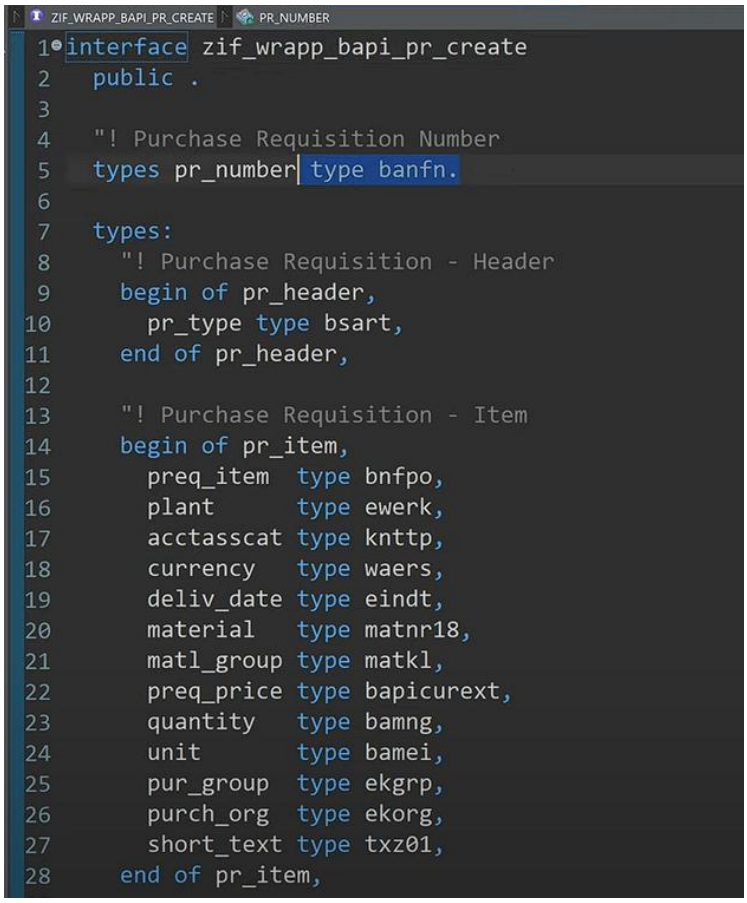
Es una capa de código que "envuelve" otra funcionalidad o componente para proporcionarle una interfaz diferente o más simplificada. Los wrappers se utilizan para adaptar el código existente a nuevos contextos o entornos sin necesidad de modificar el código original.

Interfaz como Wrapper:

En el contexto del wrapper para BAPIs, se crea una interfaz que define los métodos que envuelven las llamadas a las BAPIs. Las clases que implementan esta interfaz con la lógica para llamar a las BAPIs y manejar los datos de entrada y salida por medio de parámetros según sea necesario. Estas clases ABAP envuelven las BAPIs por medio de interfaces, permitiendo su uso en el Tier 1. Estas clases encapsulan la llamada a la BAPI y ocultan su implementación detrás de la clase ABAP.

```
FUNCTION bapi_pr_create
  IMPORTING
    VALUE(prheader) LIKE bapimereqheader OPTIONAL
    VALUE(prheaderx) LIKE bapimereqheaderx OPTIONAL
    VALUE(testrun) LIKE bapiflag-bapiflag OPTIONAL
  EXPORTING
    VALUE(number) LIKE bapimereqheader-preq_no
    VALUE(prheaderexp) TYPE bapimereqheader
  TABLES
    return LIKE bapiret2 OPTIONAL
    pritem LIKE bapimereqitemimp
    pritemx LIKE bapimereqitemx OPTIONAL
    pritemexp LIKE bapimereqitem OPTIONAL
    pritemsource LIKE bapimereqsource OPTIONAL
    praccount LIKE bapimereqaccount OPTIONAL
    praccountproitsegment LIKE bapimereqaccountprofitseg OPTIONAL
    praccountx LIKE bapimereqaccountx OPTIONAL
    praddrdelivery LIKE bapimerqaddrdelivery OPTIONAL
    pritemtext LIKE bapimereqitemtext OPTIONAL
```

Hasta que SAP libere una API específica en el Tier 1, se utiliza este wrapper como una solución temporal. Una vez que la API esté disponible, se modifica la lógica de la clase para que haga llamadas a la API estándar en lugar de la BAPI.

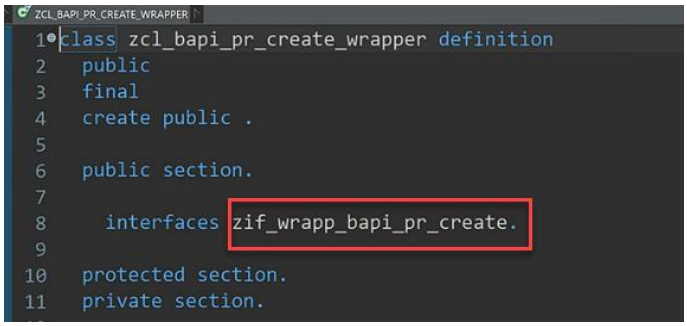


1.8. Clase Wrapper para BAPI

Continuando, explorando la habilitación de una API desde el Tier 2 al Tier 1 en el contexto de SAP S/4HANA Cloud y on-premise. Se pretende realizar esta habilitación a través de una clase wrapper que implementa una interfaz según el escenario de negocio, permitiendo así utilizar BAPIs (Business Application Programming Interfaces) en el entorno de desarrollo ABAP for Cloud Development, donde directamente no es posible llamar a BAPIs.

La idea central es crear una interfaz que defina los tipos y métodos a implementar. Un ejemplo es el método create que genera un objeto de negocio (como una requisición de compra) y tiene la capacidad de verificar usando un modo test. Luego de definir la interfaz, se debe crear una clase wrapper que envuelva la llamada a la BAPI. Esta clase se encargará de implementar los métodos definidos en la interfaz.

Ejemplo de Clase Wrapper:



Puntos importantes a considerar:

- La interfaz actúa como contrato, asegurando que cualquier clase que la implemente cumpla con ciertas funcionalidades específicas.
- La implementación de la interfaz asegura que todos los parámetros necesarios se preparen y pasen correctamente, garantizando que la llamada a la BAPI se realice de manera eficiente.
- La llamada a la BAPI se encapsula dentro de métodos privados para que no sean accesibles externamente, manteniendo así la integridad y seguridad del código.
- Los métodos privados en la clase wrapper, manejan detalles específicos de la preparación de datos antes de llamar a la BAPI.

1.9. Clase Factory Wrapper

Es una clase cuyo propósito principal es crear y devolver instancias de otra clase, en este caso, la clase wrapper que encapsula una BAPI (Business Application Programming Interface). Esta clase se utiliza para implementar el patrón de diseño Factory, que es un patrón creacional. El patrón Factory proporciona una manera de crear objetos sin especificar la clase exacta del objeto que se creará. En este contexto, permite la creación controlada y encapsulada de instancias de la clase wrapper.

La Clase Factory Wrapper proporciona de una manera estructurada y controlada de instanciar objetos de la clase wrapper que encapsula la BAPI, asegurando que todas las instancias se creen de manera consistente y siguiendo buenas prácticas de encapsulación. Esto es especialmente útil en escenarios de tier 2 API enablement, donde la

clase wrapper no puede ser utilizada directamente en el tier 1 sin pasar por un proceso de liberación adecuado.

Características Principales:

- **Estatus de liberación:** Para asegurar la correcta implementación de la clase Factory Wrapper en el tier 1, es crucial que todos los objetos utilizados en su definición e implementación, así como la clase Factory Wrapper misma, posean el estado "Released" o "Liberado". Este estatus debe estar reflejado en la sección "API State" dentro de la vista de propiedades de cada objeto.
- **Encapsulación:** La encapsulación es un concepto clave, ya que garantiza que los detalles de implementación de la clase wrapper estén ocultos. La Factory Wrapper actúa como una capa intermediaria que maneja la creación de instancias sin exponer la implementación interna.
- **Método Estático:** La clase Factory Wrapper generalmente contiene un método estático, como create_instance, que es responsable de devolver una instancia de la clase wrapper. Este método es estático porque no necesita una instancia de la clase Factory para ser llamado, facilitando así la creación de objetos de manera global y accesible.
- **Uso de Referencias de Interfaz:** En lugar de instanciar la clase wrapper directamente, la Factory Wrapper utiliza una referencia de interfaz para crear la instancia. Esto sigue el principio de programación orientada a interfaces, que promueve la flexibilidad y la reutilización del código. La interfaz define los métodos y tipos que la clase wrapper debe implementar, proporcionando una capa adicional de abstracción.
- **Visibilidad Privada y Constructor:** La clase Factory Wrapper a menudo tiene visibilidad privada para sus componentes, lo que refuerza la encapsulación. Además el método constructor de la clase wrapper puede incluir lógica adicional para manejar escenarios específicos durante la instancia, y se define en la sección privada.

Ejemplo de Clase Factory Wrapper:

```
class zcl_bapi_wrapper_factory definition
  public
  final
  create private.

  public section.

      class-methods create_instance
            returning value(result) type ref to zif_wrapp_bapi_pr_create.

  protected section.
  private section.

      methods constructor.

endclass.
```

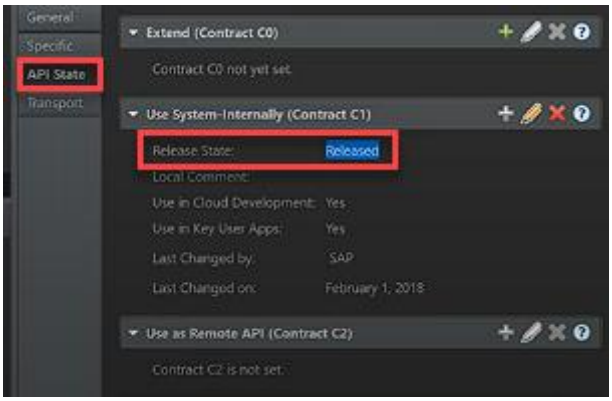
```
class zcl_bapi_wrapper_factory implementation.

  method create_instance.
     result = new zcl_bapi_pr_create_wrapper(  ).
  endmethod.

  method constructor.

  endmethod.

endclass.
```

1.10. Liberación Objetos para Tier 1

El proceso de liberación de objetos en el escenario de "Tier 2 Cloud API Enablement" se centra en permitir el acceso desde componentes de software clásicos donde reside una BAPI, encapsulada mediante la lógica de una clase Wrapper, una Factory Class y el uso de una interfaz. El objetivo es hacer disponible esta clase Factory en el entorno Tier 1, específicamente en el desarrollo de software con el componente Z ABAP Cloud, que utiliza el lenguaje ABAP for Cloud Development.

Es necesario liberar la clase Factory Wrapper que crea instancias del Wrapper, que encapsula la llamada a la BAPI. Así como la liberación de la interfaz wrapper que define los métodos que envuelven las llamadas a las BAPIs junto con los parámetros de entrada o salida.

La Wrapper Class no se debe liberar porque contiene código que no debe usarse en el Tier 1. Un ejemplo de esto sería el uso de la sentencia LIKE, CALL FUNCTION, entre otros.



1.11. Uso API Tier 2 en Tier 1

El objetivo es utilizar la lógica encapsulada en Tier 2 dentro del entorno de desarrollo Tier 1, utilizando componentes y clases del lenguaje ABAP for Cloud Development. Debido a la falta de una API oficial de SAP en esta versión del lenguaje, es necesario encapsular la lógica mediante una serie de artefactos que permiten la simulación y uso de funcionalidades a nivel de API.

No es posible en un paquete en Tier 1 con ABAP for Cloud Development, usar en Call Function en esta versión del lenguaje. Aunque se intente declarar una tabla y utilizar la BAPI, esta instrucción es obsoleta y no válida en ABAP for Cloud Development.

Por lo que es fundamental encapsular la lógica mediante la interfaz, la clase Wrapper, y la clase Factory Wrapper, liberando artefactos necesarios como se mencionó en el punto **Liberación Objetos para Tier 1** para su uso en un componente de software basado en ABAP for Cloud Development.

Además, se recomienda consultar con SAP y solicitar la liberación de una API oficial estándar que se ajuste al enfoque de negocio requerido. Como se mencionó en el punto **Tier 2 - Interfaz Wrapper para BAPI.**

1.12. API – Marcar como Obsoleta

El proceso de marcar una API como obsoleta es una práctica esencial en el desarrollo de software, que se utiliza para indicar que una API o un objeto ya no es recomendado para su uso y que será reemplazado por una versión más reciente o una alternativa mejorada. SAP aplica este proceso para sus APIs, y se recomienda que los desarrolladores hagan lo mismo con las APIs de clientes que han liberado, especialmente en escenarios de "Tier 2 Cloud API Enablement".

El uso de una API obsoleta generará advertencias en la vista de problemas (Problems), indicando que se debe usar la nueva versión.

Razones para Marcar una API como Obsoleta:

Nuevas Funcionalidades:

- La implementación de nuevas funcionalidades que cambian el proceso de negocio.
- Evitar la alteración de APIs que ya están en uso, manteniendo la estabilidad y consistencia.

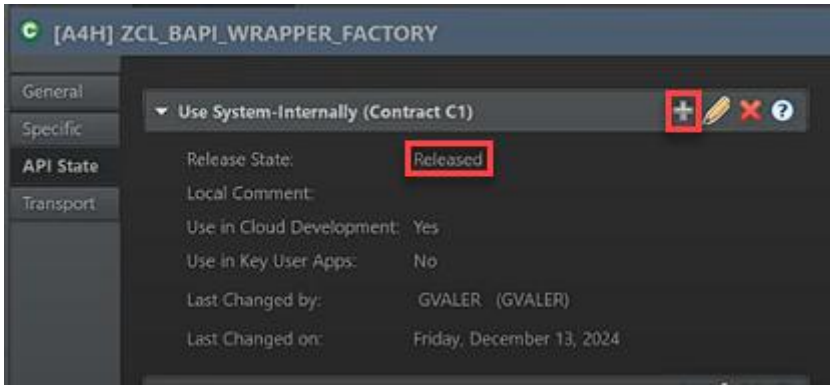
Mantenimiento y Mejora:

- Facilitar la actualización y el mantenimiento del código, evitando el uso de funcionalidades antiguas o ineficientes.
- Garantizar que los desarrolladores utilicen las últimas versiones con todas las novedades y mejoras de proceso.

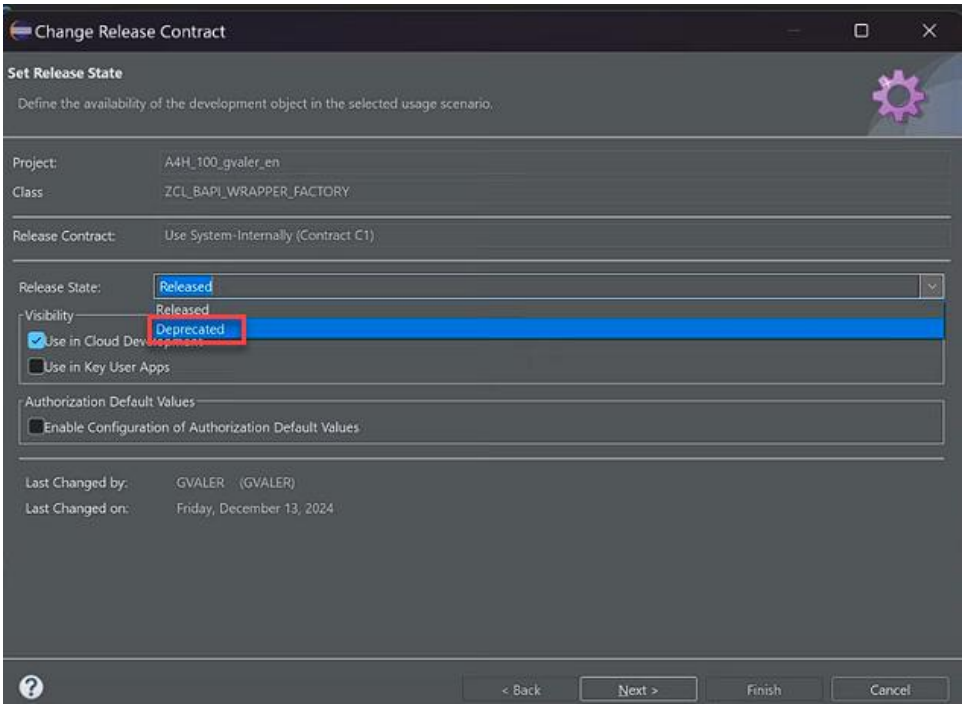
Antes de marcar una clase como obsoleta, es necesario contar con una nueva API proporcionada por SAP o crear y liberar una nueva API que la reemplazará estableciendo el estado en “Released” en la sección “API State” de la vista “Properties” .

Marcar la API Antigua como Obsoleta:

Navegar a las propiedades en la vista “Properties” de la API antigua y cambiar su estado a Deprecated en la sección “API State”. Por medio del botón con el signo “+”.



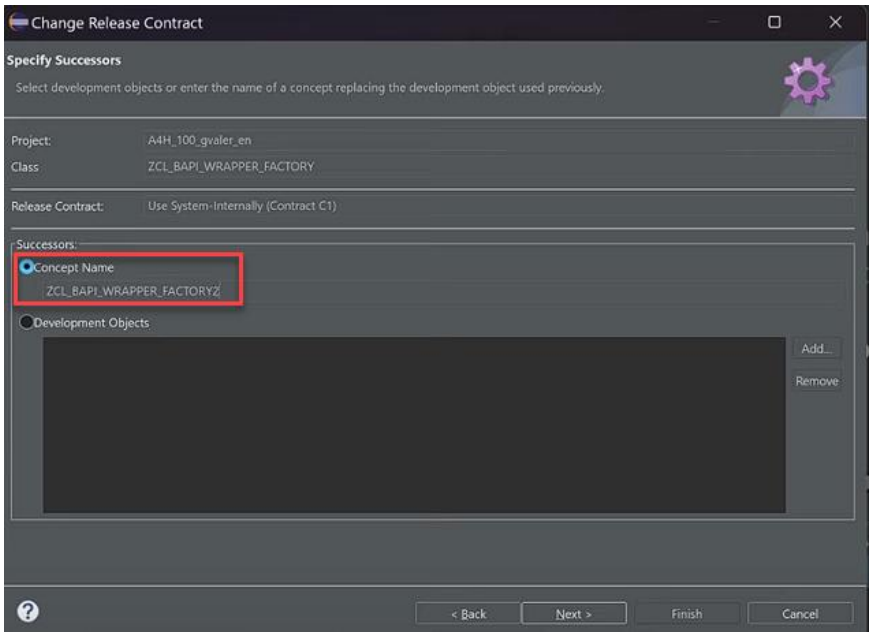
Utilizar el campo “Release State” para cambiar el estado “Deprecated". Y continuar con el proceso presionado el botón “Next”.



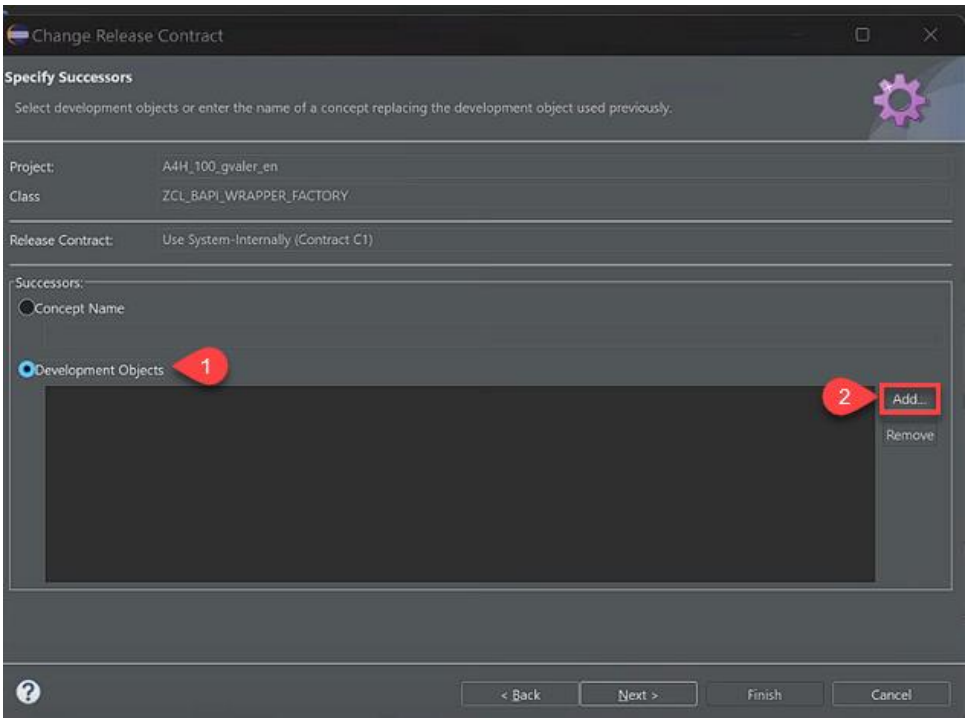
Indicar el sucesor:

Este proceso se puede hacer de 2 maneras:

- **Utilizar la opción Concept Name:** Colocando un nombre de concepto en el caso de que no se haya creado o se tenga disponible una API para la sucesión.

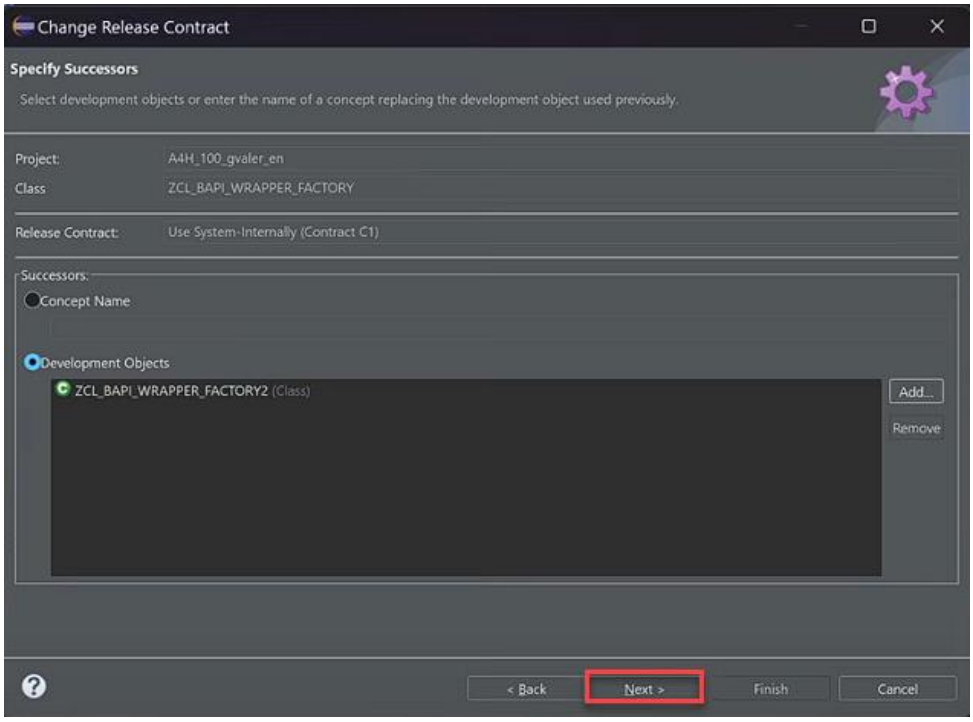


- **Utilizar la opción Development Objects:** Para referenciar el objeto sucesor en el caso de que si se poseen una API de sucesión. Para indicar el nombre del nuevo objeto o API sucesora se debe de presionar el botón “Add”.

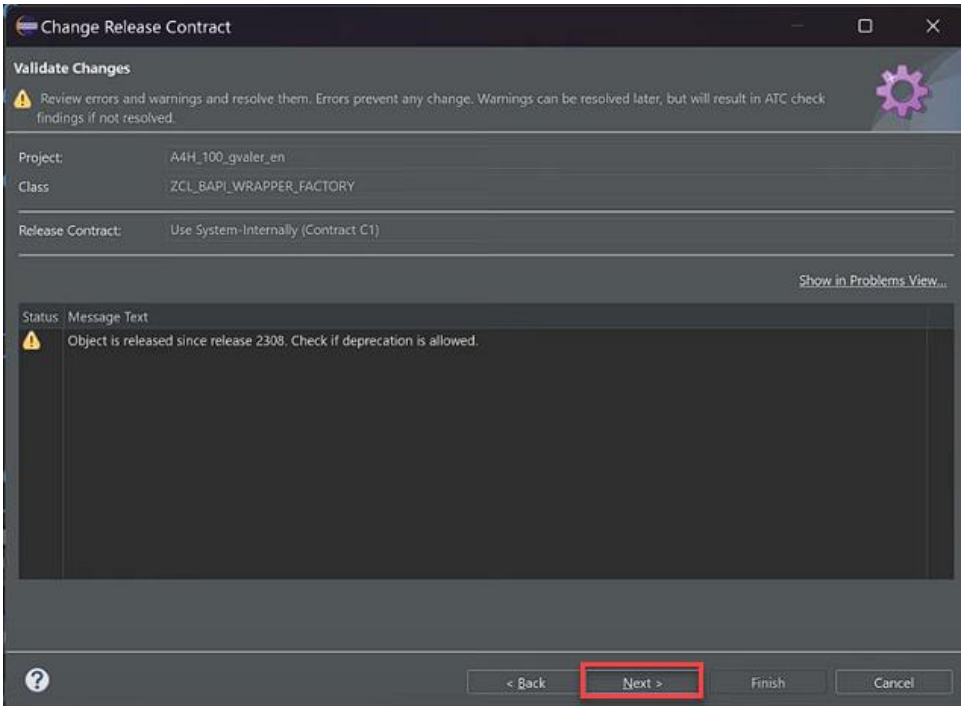


Buscar y seleccionar la API de sucesión.

Y se continúa con el proceso presionando el botón “Next”.



En la siguiente pantalla mostrará que el proceso está permitido Y se continúa con el proceso presionando el botón “Next”.



De esta manera se podrá evidenciar en la vista “Properties” de la sección “Release State” que el estado de liberación se encontrará “Deprecated” u Obsoleto.

Buenas Prácticas y Recomendaciones:

- **Consulta y Colaboración con SAP:**
  - Solicitar la liberación de APIs oficiales de SAP que se ajusten al enfoque de negocio requerido.
  - Colaborar con SAP para mantenerse actualizado con las últimas versiones y recomendaciones.
- **Encapsulación y Liberación de APIs:**
  - Encapsular la lógica de negocio de manera efectiva utilizando interfaces, clases Wrapper, y Factory.
  - Liberar estos artefactos adecuadamente para su uso en entornos de desarrollo basados en ABAP for Cloud Development.
- **Actualización del Código:**
  - Realizar revisiones periódicas del código para identificar y reemplazar el uso de APIs obsoletas.
  - Garantizar que todas las funcionalidades estén alineadas con las versiones más recientes y mejoradas.

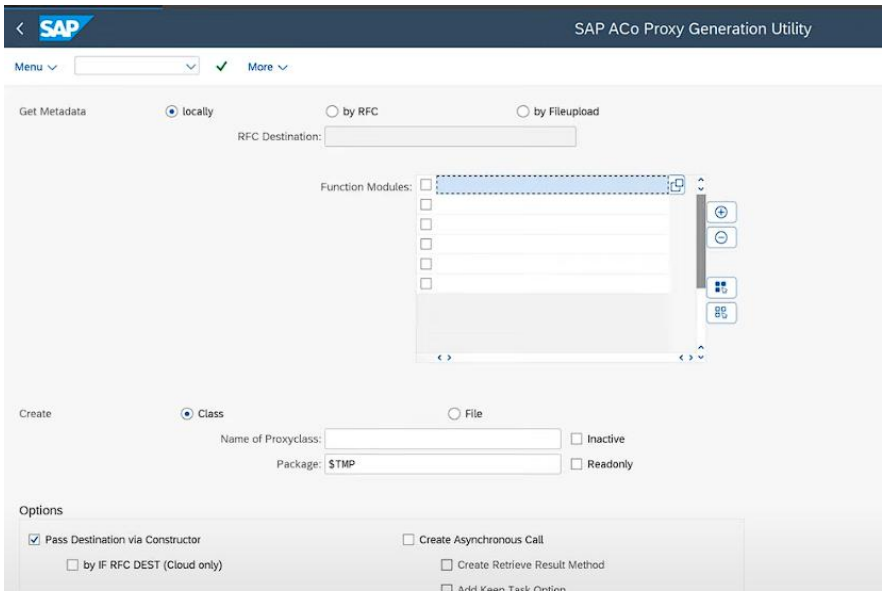
1.13. Acelerador Wrapper

La metodología del Wrapper se utiliza para encapsular una BAPI o un objeto del Tier 2 y exponerlo en el entorno del Tier 1. Este proceso puede ser laborioso dependiendo de la complejidad de los objetos a encapsular. Existen herramientas de automatización que permiten acelerar este proceso de creación de artefactos necesarios para la gestión del Tier 2 Cloud API Enablement en el Tier 1.

La encapsulación de una BAPI o un objeto del Tier 2 implica la creación de una interfaz, una clase Wrapper, y una Factory Class. Este proceso asegura que los objetos encapsulados sean accesibles y reutilizables en el entorno del Tier 1.

Herramientas de Automatización:

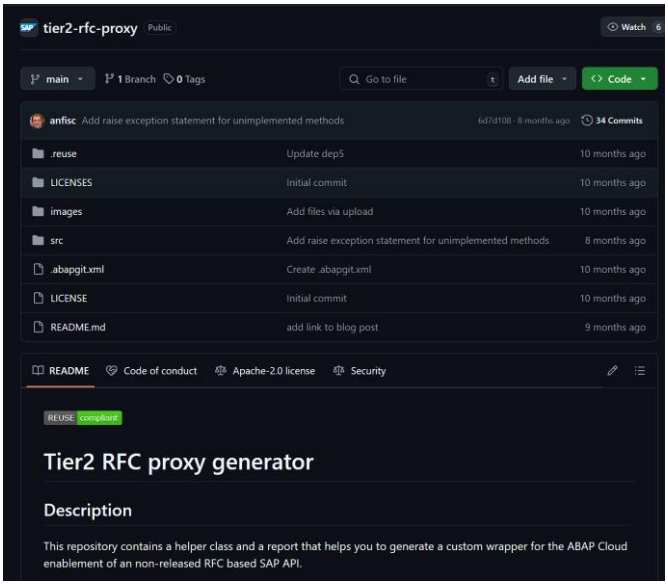
- **Transacción Estándar (ACO_PROXY)**: Es una herramienta que permite generar clases de proxy para funciones remotas (RFC) de manera eficiente. Esta transacción es especialmente útil cuando se trabaja con BAPIs (Business Application Programming Interfaces) y otros módulos de funciones remotos. La cual permite la configuración y generación de objetos Proxy basados en módulos de funciones. Esta herramienta facilita la creación de clases, paquetes de desarrollo, y excepciones basadas en clases o excepciones clásicas.



- **Tier 2 RFC Proxy Generator**: Es una herramienta que automatiza el proceso de creación de artefactos necesarios para encapsular y exponer funciones remotas (RFC) en un entorno SAP. Este generador utiliza la misma API que la transacción estándar ACO_PROXY, pero agrega funcionalidades adicionales para simplificar y acelerar el proceso.

  Fue creado por André Fischer, un contribuidor clave en el equipo de SAP Gateway y autor del RAP Generator. Este proyecto, está disponible en GitHub y se puede acceder al repositorio por medio del enlace:

  https://github.com/SAP-samples/tier2-rfc-proxy

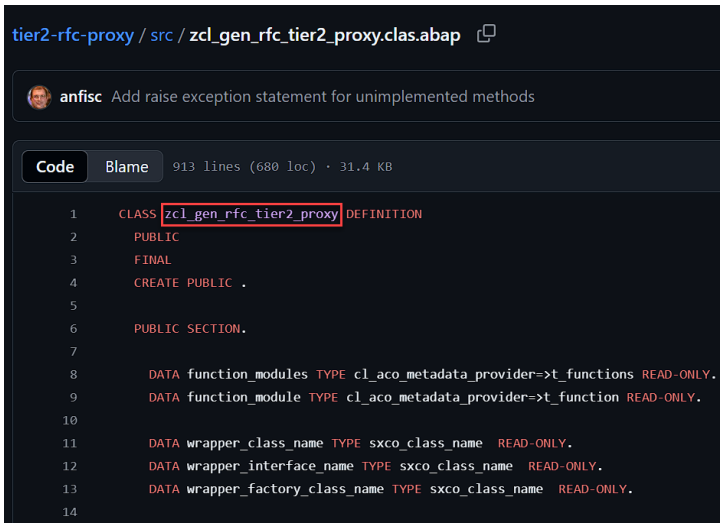


Implementación de la herramienta Tier2 RFC Proxy Generator:

Para implementar la herramienta en el sistema objetivo es necesario:

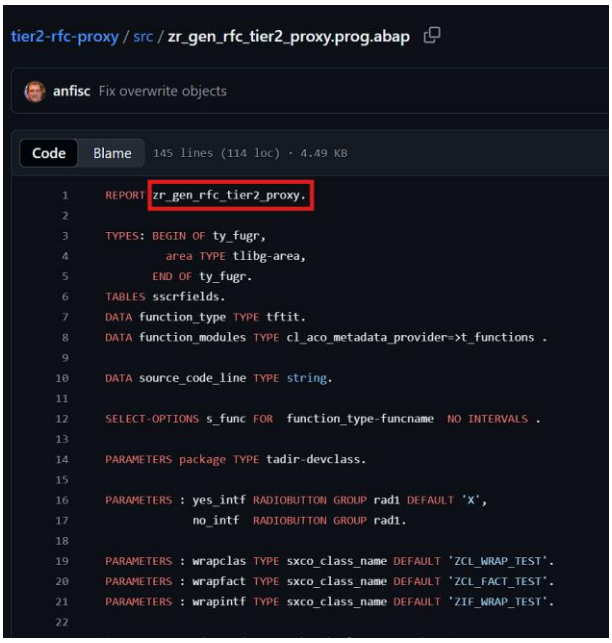
- Crear y activar la clase **ZCL_GEN_RFC_TIER2_PROXY,** el código fuente de puede ser obtenido por desde el repositorio oficial o por medio del siguiente enlace:

  https://github.com/SAP-samples/tier2-rfc-proxy/blob/main/src/zcl_gen_rfc_tier2_proxy.clas.abap



- Crear y activar la programa **ZR_GEN_RFC_TIER2_PROXY,** el código fuente de puede ser obtenido por desde el repositorio oficial o por medio del siguiente enlace:

  https://github.com/SAP-samples/tier2-rfc-proxy/blob/main/src/zr_gen_rfc_tier2_proxy.prog.abap



Al ejecutar el **ZR_GEN_RFC_TIER2_PROXY,** se puede especificar una función bapia ha encapsular, el paquete de desarrollo. Luego es posible seleccionar las opciones:

- **Inf. class and fact. class:** Para crear la estructura descrita en el desarrollo completo de esta documentación como lo serían la interfaz Wrapper, Clase Wrapper y Clase Factory Wrapper.

- **Only wrapper class:** Es posible crear solamente una Clase Wrapper con la implementación de los métodos que llamarán la bapi. Aunque es recomendable utilizar la otra opción para mantener la estructura correcta de los artefactos de encapsulación

Para luego colocar los nombres de los diferentes elementos a ser creados como interfaz Wrapper, Clase Wrapper y Clase Factory Wrapper.

Por último la opción "Patch Obj" se utiliza para actualizar o parchar los objetos generados previamente. Esta opción permite realizar modificaciones incrementales en los objetos sin necesidad de regenerarse completamente desde cero. Por lo que es recomendable no seleccionar dicha opción y colocar objetos que no hayan sido creados para su creación por medio de la herramienta.