PUBLIC

ABAP Cloud API Enablement Guidelines for SAP S/4HANA Cloud, private edition, and SAP S/4HANA

Guidelines for administrators and ABAP developers

VERSION 1.0 / MAY 2023

Table of Contents

1 INTRODUCTION

1.1 Overview of the Three Tier Model

The new ABAP Extensibility Guide introduces the 3-tier model bringing the ABAP cloud development model to SAP S/4HANA Cloud, private edition and SAP S/4HANA. The overall idea is depicted below.

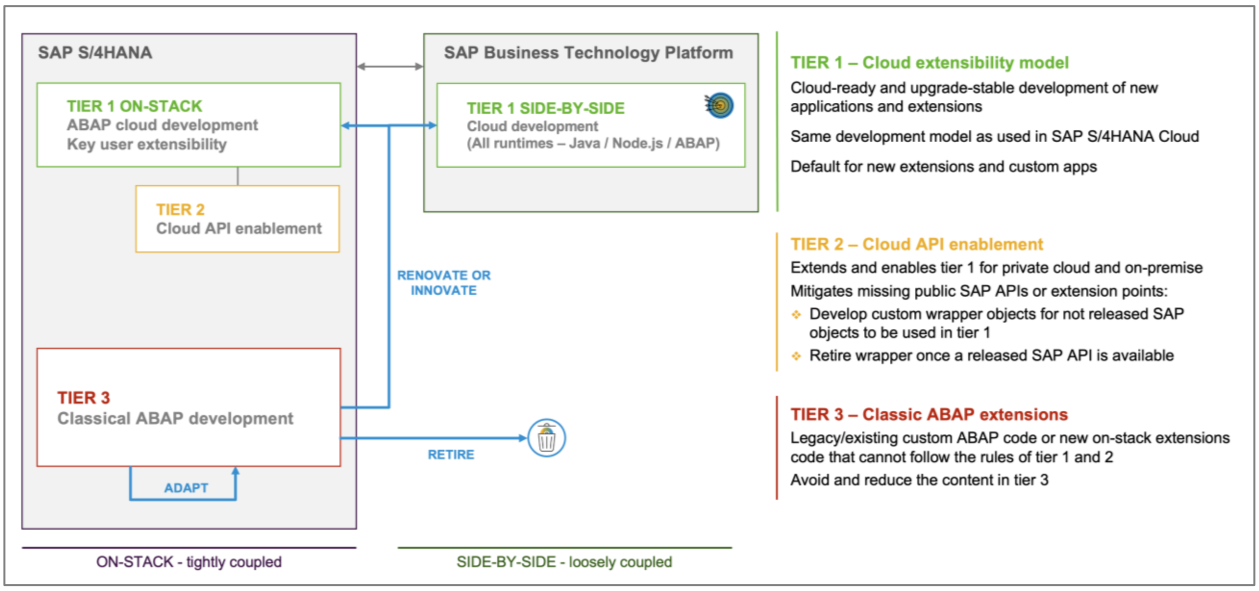
SAP S/4HANA
TIER 1 ON-STACK
ABAP cloud development
Key user extensibility
TIER 2
Cloud API enablement
TIER 3
Classical ABAP development
ADAPT
SAP Business Technology Platform
TIER 1 SIDE-BY-SIDE
Cloud development
(All runtimes – Java / Node.js / ABAP)
RENOVATE OR INNOVATE
RETIRE
ON-STACK - tightly coupled
SIDE-BY-SIDE - loosely coupled
TIER 1 – Cloud extensibility model
Cloud-ready and upgrade-stable development of new applications and extensions
Same development model as used in SAP S/4HANA Cloud
Default for new extensions and custom apps
TIER 2 – Cloud API enablement
Extends and enables tier 1 for private cloud and on-premise
Mitigates missing public SAP APIs or extension points:
Develop custom wrapper objects for not released SAP objects to be used in tier 1
Retire wrapper once a released SAP API is available
TIER 3 – Classic ABAP extensions
Legacy/existing custom ABAP code or new on-stack extensions code that cannot follow the rules of tier 1 and 2
Avoid and reduce the content in tier 3

Figure 1.1 - The 3-tier extensibility model

The 3-tier extensibility model allows to separate extensions following the cloud extensibility model (tier 1) and existing custom code or legacy developments using classic extensibility techniques (tier 3). In between, there is tier 2 as cloud API enablement layer that becomes important if a public SAP API is missing.

The figure below illustrates the interplay of the three tiers and how they access SAP development objects:

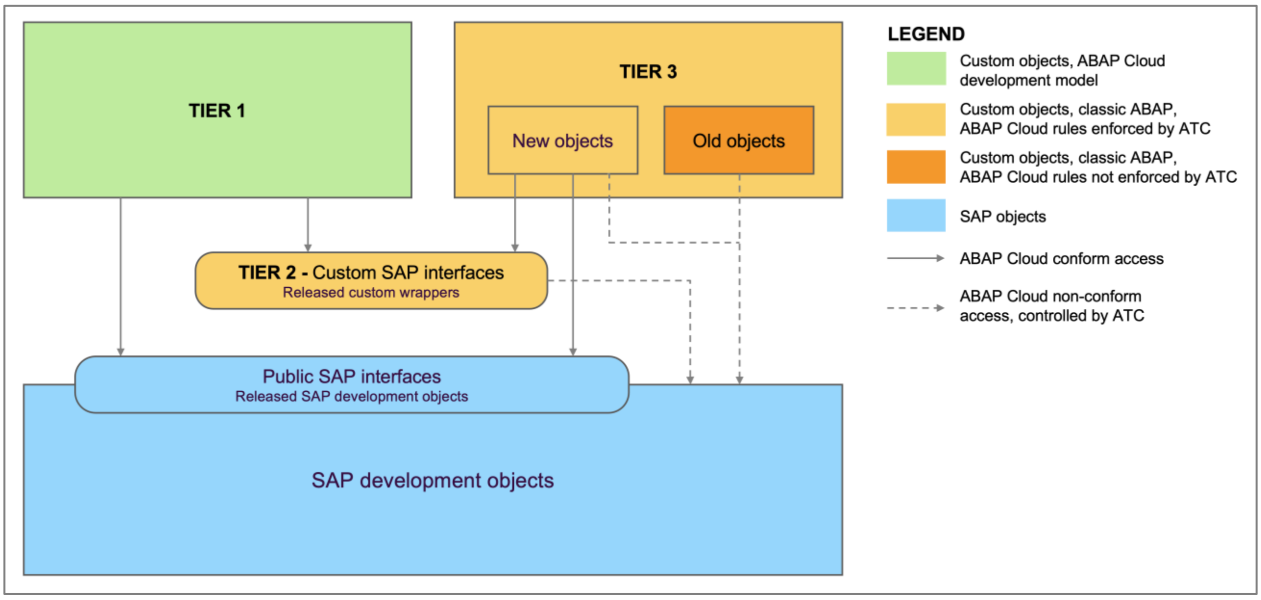
TIER 1
TIER 3
New objects
Old objects
TIER 2 - Custom SAP interfaces
Released custom wrappers
Public SAP interfaces
Released SAP development objects
SAP development objects
LEGEND
Custom objects, ABAP Cloud development model
Custom objects, classic ABAP, ABAP Cloud rules enforced by ATC
Custom objects, classic ABAP, ABAP Cloud rules not enforced by ATC
SAP objects
ABAP Cloud conform access
ABAP Cloud non-conform access, controlled by ATC

Figure 1.2 - Interplay of the three tiers

The custom code in tier 1 follows ABAP Cloud rules and hence by default can only access the public SAP interfaces. They are given by the SAP development objects that are released for cloud development (see also the blog post “Embedded Steampunk – Some more details…”). In case APIs for business logic or data access are missing in the public SAP interfaces, tier 2 becomes relevant: it comprises wrapper objects developed in classic ABAP, which access the non-released SAP development objects, and are released for ABAP Cloud[1] to be usable in tier 1. In this way, tier 2 constitutes custom interfaces to SAP development objects. Access to the released development objects in the public and the custom SAP interfaces is ABAP Cloud conform (solid arrows), while access to non-released SAP development objects is not ABAP Cloud conform (dashed arrows). If applicable, for tier 3 it is also recommended to use the public or custom SAP interfaces of tier 2 for new developments. While for tier 1 the ABAP Cloud rules are enforced via syntax and runtime checks, the ABAP Cloud conformity of tier 2 and tier 3 shall be controlled by the ABAP test cockpit.

In this guide, we provide more information regarding the setup of tier 2, about building wrappers for non-released SAP objects and about consuming these wrappers in tier 1.

1.2 What's In?

The purpose of tier 2 is to mitigate missing public SAP APIs while sticking to the ABAP cloud development rules and guidelines as much as possible.

With tier 2 you can

- mitigate missing public SAP APIs that are required for ABAP cloud development.
- control the dependencies to non-released SAP development objects by building a clear custom interface between SAP and custom code.

1.3 Expectations and Obligations

Although objects that are suited for building wrappers to mitigate missing public APIs are almost stable, there is no guarantee for upgrade-stability as we have for the public SAP interfaces. Hence, anything developed in tier 2 needs to be reviewed and tested during upgrade projects.

To offer a broad scope of mitigation possibilities in tier 2, possible violations of strict ABAP Cloud rules in legacy objects need to be tolerated. As a consequence, it is possible that a tier 1 development using a tier 2 wrapper does not fulfill all qualities as a pure tier 1 development (e.g. regarding transactional consistency, see the blogs: The SAP LUW in ABAP Cloud and Using BAPIs in RAP). It is the obligation of the customer to ensure correct functionality in these scenarios. SAP offers tools to identify possible issues.

The guidelines for tier 2 given in this paper are not enforced by SAP, but they are a recommendation that may be adapted to meet future needs. It is the customer's responsibility to establish these or similar guidelines in his company.

[1] Short notation for releasing the object under the contract C1 with visibility "Use in Cloud Development".

2 DEVELOPMENT GUIDELINES FOR TIER 2

In this chapter some recommendations regarding the technical setup of tier 2 are provided, followed by guidelines regarding the development of wrappers. Finally, proposals for SAP APIs to be wrapped are made.

2.1 Setup Recommendations

The basic setup of the 3-tier Model is already explained in the ABAP Extensibility Guide. Here, we summarize the setup of tier 2.

2.1.1 Software Component

In tier 2 access to non-released SAP objects is required. Hence, the created objects must be in a software component with default language version Standard ABAP[2] and cannot be in the same software component as tier 1. It is not required to create a software component solely for tier 2, but the recommendation is to use packages in HOME or an existing software component with default language version Standard ABAP.
Note: For development within a defined customer or partner namespace however, it is possible to create a separate software component for tier 2.

2.1.2 Packages

It is recommended to create dedicated packages or a package tree containing all objects required for building wrappers. The transport layer for tier 2 packages must correspond to the transport layer of the tier 1 developments to ensure that the wrappers are transported along the same transport routes as the objects using them.

2.1.3 ABAP Test Cockpit

Tier 2 shall be as close to ABAP Cloud as possible, meaning that all objects and coding should be created as it would be required in tier 1, except for single dedicated violations. Hence it is recommended to check for any violation of strict ABAP Cloud principles using ABAP Test Cockpit (ATC). Two ABAP Test Cockpit check variants are provided to establish a governance regarding tier 2 development:

- **ABAP_CLOUD_DEVELOPMENT_DEFAULT**:
  With this check variant the basic ABAP Cloud rules can be enforced. Moreover, by the check category *API Release* included in this check variant a consistent release of the wrappers for ABAP Cloud, and stability in case of changes of the wrappers can be ensured by enforcing the contract rules. Further information on API snapshots required for the stability of wrappers is given in the documentation.
- **ABAP_CLOUD_READINESS**:
  The check category *Cloud Readiness* included in this check variant allows you to enforce that only object types available in ABAP Cloud are developed, only ABAP Cloud language statements are used, and that only released SAP APIs are called. Versioned lists of public APIs can be found in github.

As explained before, the intent of the wrappers in tier 2 is to constitute the custom interface to non-released SAP objects. This will hence lead to ABAP Test Cockpit errors with respect to the *Cloud Readiness* checks for which exemptions need to be requested. Quality managers can apply governance to these exemptions and thereby control the violations of ABAP Cloud and clean core rules. If public SAP APIs are provided and replace non-released objects that are used in wrappers, the ABAP Test Cockpit will detect this and indicate refactoring options.

More information on how to combine the above ABAP Test Cockpit variants in a custom variant to optimally serve this governance approach for all developments done in classic ABAP, including wrapper development in tier 2, is provided in the extensibility guide.

[2] *Standard ABAP* is the technical name of the language version in the classic ABAP development model.

2.1.4 Authorizations

It is recommended to create a dedicated new developer role for developers that are authorized and skilled to develop mitigations for missing SAP APIs in tier 2. A good starting point is to use the role template for tier 1 (SAP_BC_ABAP_DEVELOPER_5) and add the authorization for classic ABAP. This is done using the authorization object S_ABPLNGVS. More information can be found in the documentation. The role template includes the authorizations for all object types that are part of ABAP Cloud and tier 1. Further object types are not required for tier 2 development[3].

2.1.5 Development environment

ABAP development tools in Eclipse (ADT) must be used for tier 2 development. Several modern development object types like CDS View Entities, Behavior Definitions and others can only be edited in ADT.

2.2 Basic Principles

The general idea for developing wrappers is:

- Create wrapper objects in classic ABAP in tier 2.
- Implement the objects and make use of non-released SAP development objects.
- Release the required objects for ABAP Cloud*.* These objects form your custom SAP interface.

In addition to the governance possibilities explained above, the wrappers encapsulate the usage of SAP development objects, which can ease later refactoring and allow developers to concentrate test efforts to dependencies that are potentially not upgrade-stable during SAP software upgrades.

In general, the development in tier 2 shall be as compatible to ABAP Cloud as possible. That means that only object types available in ABAP Cloud shall be developed and the implementation shall follow ABAP Cloud syntax rules. If set up as recommended before, any violation of that will result in ABAP Test Cockpit errors for which dedicated exemptions need to be requested.

Finally, the need to develop a wrapper can indicate that a public SAP API is missing. SAP encourages customers and partners to make use of the Customer Influence channel for released public APIs to request the missing API.

2.3 Building Wrappers

2.3.1 General Aspects

2.3.1.1 ABAP Dictionary Artefacts

SAP does not release all ABAP Dictionary artefacts (Domains, Data Elements, Structures, Table Types, ...). However, where types or structures are of central importance, or if they are used to type the key fields of released CDS views, these types should be released and reused in custom development. If such a dictionary artefact is missing, it should be requested via the Customer Influence channel. Still, the question arises of how to deal with non-released dictionary artefacts when building wrappers in tier 2. The recommendations are:

- If the types are only required in the business logic for calling the SAP API and in tier 1 for calling the wrapper, then the types should be defined locally as part of the wrapper. This can be done using the TYPES statement in wrapper classes or interfaces (an example is provided in the Extensibility Guide). This way the wrapper class or interface is self-contained, no further dictionary artefacts need to be created or released for the usage of the wrapper in tier 1.
- If the types are required in tier 1 to define further ABAP Dictionary artefacts (Data Elements, Tables, …) they should be recreated as custom dictionary artefacts in tier 1 directly[4].

[3] Authorizations for specific object types are steered by authorization object S_DEVELOP.
[4] The dictionary artefacts created in tier 1 can be used in tier 2 if they are released for cloud development.

ABAP Dictionary search helps are not part of ABAP Cloud and will not be released by SAP. Instead, search helps are provided by corresponding CDS view entities with the required annotations. If such a released value help view entity is missing, the recommendation is to create a custom value help view on top of the corresponding table as described in the next section.

2.3.1.2 Authorizations

If PFCG roles are created, the role administrator needs to know which authorizations are required for the desired activities. While all objects released for ABAP Cloud have a clear authorization signature and the corresponding authorization objects are released, this does not necessarily hold true anymore if a wrapper of tier 2 is used. In wrapped non-released SAP APIs, there can be authorization checks on objects that are not released. If known, the recommendation is to provide corresponding authorization default values (transaction SU22) for wrappers when releasing the wrapper object for ABAP Cloud. In addition, the developer should create for each tier 1 service (that uses non-released authorization checks at runtime) a Variant for Default Data in SU22 with the non-released authorization objects from wrappers of tier 2.

Later, role administrators can construct PFCG roles for tier 1 extensions based on the authorization default values of public SAP APIs used in tier 1 and their corresponding default variants in tier 2. Authorization objects themselves (your own authorization objects in tier 3 or non-released SAP authorization objects) are not wrapped and do not need to be released for ABAP Cloud. Moreover, PFCG roles are customizing and are not part of the 3-tier model.

2.3.1.3 Connectivity

To consume HTTP services, SOAP services or RFC enabled function modules in tier 1 developments, it is required to access the corresponding SM59 destination or logical port. Currently, there are no released public APIs available for that. Instead, we recommend to write different wrapper classes around CL_OUTBOUND_PROVIDER_HTTP (or _RFC, or _SOAP) for each use case. More information is provided in this documentation. A released API for this functionality is planned for future releases.

2.3.1.4 Transactional Consistency

For wrappers in tier 2 it is possible to call SAP APIs and use ABAP Language statements that are not allowed in strict ABAP Cloud rules in tier 1. In pure tier 1 development these violations would result in syntax or runtime errors. If these rules are violated in wrappers however, they are tolerated because it cannot be guaranteed that all legacy APIs used in the wrappers follow the ABAP Cloud rules[5]. Consequently, it is the responsibility of the tier 2 developer together with the tier 1 developers to ensure correct functionality of the wrappers in tier 1 developments. In the last chapter of this paper, it is shown how to detect possible pitfalls.

2.3.2 Mitigate Missing Business Logic and Data Access APIs

The following non-released objects may be wrapped to mitigate missing public APIs:

TYPE OF SAP OBJECT	STEPS TO CREATE A WRAPPER
RAP[6] BUSINESS OBJECT	• Create a self-contained wrapper class (details below). • Provide methods for each access to the RAP BO and call them programmatically via Entity Manipulation Language (EML). • Release the developed class for ABAP Cloud and define SU22 values.
FUNCTION MODULE CLASS / INTERFACE	• Create a self-contained wrapper class (details below). • Provide methods for each access of a non-released SAP function module or class/interface and call it in the implementation.

[5] The ABAP statements COMMIT WORK and ROLLBACK WORK are never tolerated in a strict transactional context and will always lead to a runtime error, also if they are executed in a wrapper.
[6] Short for ABAP RESTful application programming model

	• Release the developed class for ABAP Cloud and define SU22 values.
CDS VIEW / TABLE	• Create a CDS View Entity on top of the non-released SAP table or view. Note: • Prefer CDS View Entities over tables. Dictionary views are not supported. • Prefer use-case specific CDS views instead of complex ones. • Decouple the CDS View Entity wrapper from underlying CDS annotations by *@Metadata.ignorePropagatedAnnotations: true*. Re-insert only the annotations required in the wrapper. • Create a DCL for the newly created CDS View Entity inheriting from the SAP DCL or defining own authorization checks if required. • Release the developed CDS View Entities for ABAP Cloud and define SU22 values.

Table 2.1 - Non-released objects that can be wrapped to mitigate missing public APIs.

For the development of wrapper classes, we recommend:

- Declare relevant types locally to decouple from non-released dictionary artefacts (see recommendation regarding ABAP dictionary artefacts above).
- Use class-based exceptions inheriting from CX_STATIC_CHECK, use factory methods instead of public constructors, declare wrapper classes as FINAL.
- Follow Clean ABAP Guidelines.

It is good practice to declare the public types and methods in an interface. Note that when releasing the wrapper class for tier 1 development, consistency of the wrapper is enforced via ABAP test cockpit. Hence, along with the wrapper class the corresponding exception class as well as referenced types or interfaces must be released. Information on how to release development objects for ABAP Cloud can be found in the documentation. Once a wrapper is released, it must only be changed compatibly in order not to harm consuming applications. To that end the API snapshots and corresponding ABAP Test Cockpit checks must be used (as explained in above section), to enforce the contract rules.

2.3.3 Mitigate Missing SAP HANA Features

Besides usual ABAP CDS View Entities it is possible to develop CDS Table Functions or ABAP Managed Database Procedures (AMDP) to work natively with SAP HANA and benefit from code-pushdown. However, in tier 1 only a limited scope is released (READ-ONLY). In tier 2, the features that are not yet offered in ABAP Cloud can be mitigated. This mitigation is not achieved by a wrapper of an SAP API, but by developing the required object in tier 2 and releasing it for ABAP Cloud.

USAGE IN TIER 1 AS:	STEPS
CDS TABLE FUNCTION	• Create the CDS Table Function in tier 2. • Create the implementing ABAP Class in tier 2 and implement the required interface using all desired SAP HANA features. • Create a CDS View Entity using the CDS Table Function. • Release the CDS View Entity for ABAP Cloud.
AMDP METHOD	• Create the class implementing the AMDP method in tier 2.

	• Implement the methods using all desired SAP HANA features. • Release the class for ABAP Cloud.

Table 2.2 - Usage of code pushdown capabilities in tier 1

The released objects can be used without any restrictions in tier 1.

2.3.4 Further Remarks

2.3.4.1 Analytics

In ABAP Cloud development we have analytical models purely based on CDS. Classic Business Warehouse (BW) models are not part of ABAP Cloud. In tier 2 it is not possible to wrap classic BW models to enable them for CDS based analytics in tier 1. That is because BW models are generated in each system, whereas CDS-based analytical models are developed in the development system and are transported through the landscape. This disjoint lifecycle of the two concepts makes it impossible to manage dependencies safely. That's why classic BW models are always located in tier 3. On the other hand CDS-based analytical models are located in tier 1. If required CDS based analytical models are not released, they can be wrapped as described above.

2.3.4.2 Objects released for key user extensibility but not for developer extensibility in ABAP Cloud

Objects released for key user extensibility are also upgrade-stable. They follow the same stability contract as for developer extensibility in ABAP Cloud. There are objects that are released for key user extensibility only, but not for developer extensibility in ABAP Cloud. This can occur for example, in the following domains:

- Analytical queries[7]:
  Released analytical queries can be used in analytical key user extensibility scenarios only. There is no use case of re-using analytical queries in ABAP cloud development. Customers can create their own analytical queries based on the underlying released CDS views (base or cube views).
- CDS views for data replication scenarios[8]:
  These views are for use in data replication tools and must not be used in ABAP cloud development.
- CDS views for forms and email output[9]:
  These views are for use in output management key user tools and must not be used in ABAP cloud development.

Note that the list is not complete, it contains the most important domains and cases only. If an object not falling under these categories is missing a release for ABAP cloud development, a corresponding requirement shall be opened in the Customer Influence channel.

2.3.4.3 Non-released BAdIs

Non-released BAdIs shall not be implemented in tier 2 but are part of tier 3. If for the implementation of a released BAdI in tier 1 an API is missing, a wrapper in tier 2 around that API can be used.

2.4 Objects Suited to be Wrapped

In general, only objects with API character shall be wrapped, but not just any SAP framework code or internal form routines. Suitable objects are for example:

[7] Identified by annotation *@ObjectModel.ModelingPattern = ANALYTICAL_QUERY* or *@Analytics.query: true*.
[8] Can be identified by the respective CDS annotations, e.g., *@ObjectModel.supportedCapabilities: #EXTRACTION_DATA_SOURCE*).
[9] Can be identified by the respective CDS annotations, e.g., *@ObjectModel.ModelingPattern: #OUTPUT_EMAIL_DATA_PROVIDER* or *#OUTPUT_FORM_DATA_PROVIDER*.

- BAPI function modules (see transaction BAPI). Information about the usage of BAPIs in a tier 1 RAP BO is provided in this Blog Post.
- Externally released function modules.
- Non-released RAP BOs (look for objects of type BDEF in ADT).
- CDS views or tables, with preference for CDS views.

Further recommendations regarding which objects are suited to be wrapped are in preparation.

3 CONSUMPTION IN TIER 1

In this chapter it is explained how to use wrappers in tier 1, how to detect issues in wrappers that might have impact on the correct functionality of your tier 1 development and how to refactor if a public SAP API is made available.

3.1 Usage

Wrapper objects of tier 2 can be used in tier 1 in the same way as public SAP APIs, since they are released for ABAP Cloud.

To determine and grant the necessary authorizations, role administrators can use comparison with SU22 data (SU25) to copy the SU22 data delivered by the developer with the release of the wrapper and the default variant into the customer-specific tables (SU24 data). See From the Programmed Authorization Check to a Role | SAP Help Portal (transaction SU25 step 2a and 2b). This way, role administrators can construct PFCG roles for tier 1 extensions based on the authorization default values of public SAP APIs used in tier 1 and their corresponding default variants in tier 2.

Additional information: For default variants it is not possible to set the check indicator: *Do not check*. This is only possible when maintaining the authorization default values. If a non-released authorization object shall not be checked for your tier 1 service, then add the corresponding authorization object to the authorization default values of the service in the SU24 transaction and set the check indicator: *Do not check*.

3.2 Error Logs

There will be no syntax or runtime errors if wrappers violate strict ABAP Cloud rules as explained before. Instead, violations can be logged and used to detect possible issues and improve the wrappers and their usage.

- Any violation of transactional consistency rules of the SAP LUW will be logged via checkpoint group CC_STMT.
- This checkpoint group can be activated using transaction SAAB and the recorded logs can be inspected.

There are two recommended options:

- Activation of the checkpoint group for dedicated tests for specific users and time frames (that could e.g., be part of done criteria for tier 1 developments) and analysis of all findings reported in the logs, including findings inside wrappers.
- Permanent activation of the checkpoint group in the development system, regular check upon any errors by developers.

3.3 Replacement and Refactoring

After an SAP upgrade, the ABAP Test Cockpit checks in check category *Cloud Readiness* can be used to identify refactoring possibilities (see setup recommendations above). If a public SAP API is available for a wrapped functionality, the public released SAP APIs should be used instead of unreleased objects. For refactoring in this regard, there are two options:

- Refactoring in tier 1:
  - Replace the usage of the wrapper with direct usage of the public SAP API.
  - Adapt PFCG roles to include the authorization objects of the public SAP APIs instead of the ones required for the wrapper.
  - Delete the wrapper in tier 2 if it is not used any more.
- Refactoring in tier 2:
  - Replace the usage of the non-released object in the wrapper by the released SAP API.
  - Adapt the SU22 values provided when releasing the wrapper for cloud development and the default variants to only include the released authorization objects of the released SAP APIs.
  - Adapt the PFCG roles accordingly.

- Change the ABAP language version of the wrapper object to ABAP for Cloud Development, remove any violation of strict ABAP Cloud rules.
- Move the wrapper to the tier 1 software component by assigning it to the corresponding tier 1 development package[10].

Which option to follow, must be decided case-by-case. If the wrapper object is used multiple times in tier 1 (see e.g., where-used list), option 2 can be suitable, because it minimizes the refactoring effort and keeps a central and reusable access point to SAP APIs. For single or simple usages option 1 can be appropriate as well.

Like in every refactoring project, careful continuous testing during the entire refactoring project is required. Here, this also includes the usages of the wrapper or SAP APIs in tier 1.

[10] The package reassignment must be on a separate transport request. The required authorizations (S_CTS_ADMI) are not included in the authorization recommendation for tier 2 as explained above.