UX403

SAP Fiori Elements Development

PARTICIPANT HANDBOOK
INSTRUCTOR-LED TRAINING

Course Version: 22
Course Duration: 5 Day(s)
Material Number: 50159303

SAP Copyrights, Trademarks and Disclaimers

© 2022 SAP SE or an SAP affiliate company. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP SE or an SAP affiliate company.

SAP and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP SE (or an SAP affiliate company) in Germany and other countries. Please see https://www.sap.com/corporate/en/legal/copyright.html for additional trademark information and notices.

Some software products marketed by SAP SE and its distributors contain proprietary software components of other software vendors.

National product specifications may vary.

These materials may have been machine translated and may contain grammatical errors or inaccuracies.

These materials are provided by SAP SE or an SAP affiliate company for informational purposes only, without representation or warranty of any kind, and SAP SE or its affiliated companies shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP SE or SAP affiliate company products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.

In particular, SAP SE or its affiliated companies have no obligation to pursue any course of business outlined in this document or any related presentation, or to develop or release any functionality mentioned therein. This document, or any related presentation, and SAP SE's or its affiliated companies' strategy and possible future developments, products, and/or platform directions and functionality are all subject to change and may be changed by SAP SE or its affiliated companies at any time for any reason without notice. The information in this document is not a commitment, promise, or legal obligation to deliver any material, code, or functionality. All forward-looking statements are subject to various risks and uncertainties that could cause actual results to differ materially from expectations. Readers are cautioned not to place undue reliance on these forward-looking statements, which speak only as of their dates, and they should not be relied upon in making purchasing decisions.

Typographic Conventions

American English is the standard used in this handbook.

The following typographic conventions are also used.

This information is displayed in the instructor's presentation	
Demonstration	
Procedure	
Warning or Caution	
Hint	
Related or Additional Information	
Facilitated Discussion	
User interface control	*Example text*
Window title	*Example text*

© Copyright. All rights reserved.

Contents

© Copyright. All rights reserved.

© Copyright. All rights reserved.

Course Overview

TARGET AUDIENCE

This course is intended for the following audiences:

- Developer
- Solution Architect
- User Experience Designer

© Copyright. All rights reserved.

© Copyright. All rights reserved.

UNIT 1 SAP UX Strategy

UNIT OBJECTIVES

- Describe SAP User Experience Strategy
- Explain SAP User Experience Tools and Technologies
- Describe SAP User Experience Use Case for Building Fiori-like Apps

© Copyright. All rights reserved.

© Copyright. All rights reserved.

Unit 1
Lesson 1

Describing SAP User Experience Strategy

LESSON OBJECTIVES
After completing this lesson, you will be able to:

- Describe SAP User Experience Strategy

User Experience Strategy

Scenario

As a Developer, you are requested by the business process owners to permanently improve the user experience.

Creating user interfaces is a critical aspect in the user experience. In this training, you will extend your knowledge in developing an SAPUI5 user interface.

The World is Changing

Figure 1: The World is Changing

For example, if a user is purchasing something personal online like a television, that same user would expect that if they needed to order something from work, say a new cell phone, that the process would be the same in terms of ease and simplicity. This idea has set the bar for expectations when using business applications.

Consumerization is the growing tendency for new information technology to emerge first in the consumer market and then spread into business and government organizations.

This is evidenced today more so than ever before. People are used to the application user experience they interact with their daily applications such as Google and social media apps.

This experience has been embraced and accepted by people to the point where it is now the new standard. People want to have the same easy, feature-rich experience they have with their personal sites in the business place.

© Copyright. All rights reserved.

SAP has recognized this change in society and made it our goal to meet this new standard to transform the user experience.

Digital Transformation — The Influence on User Expectations

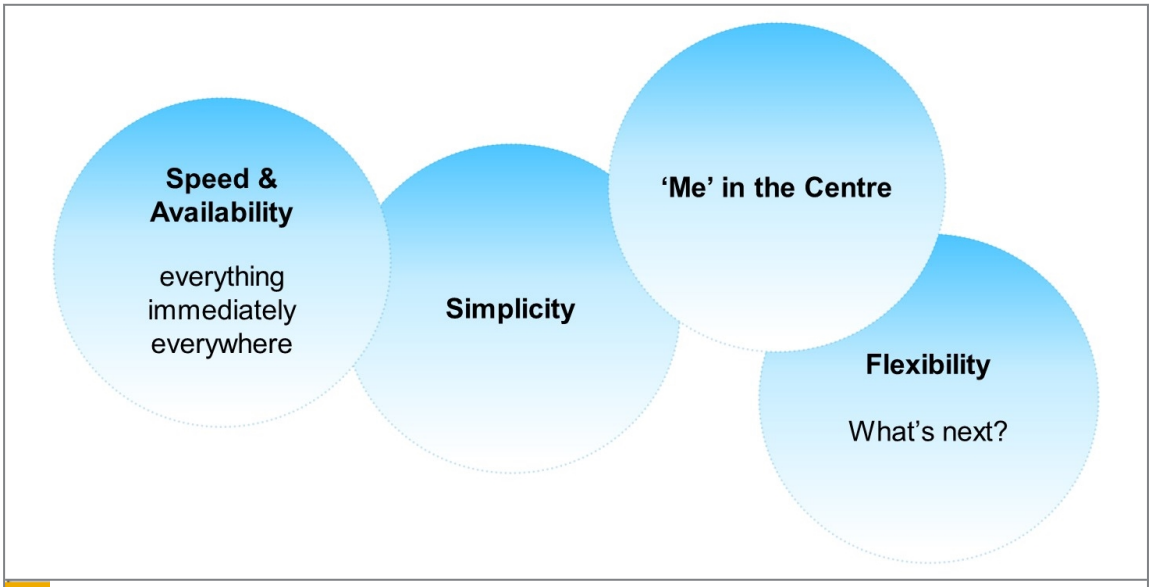


Figure 2: Digital Transformation — The Influence on User Expectations

SAP developed a user experience strategy consisting of three components:

- New
- Renew
- Enable

During our research phase, we realized that most users still use the SAP GUI to access applications.

The GUI contains approximately 300,000 screens and consists of a vast number of functionalities.

We looked at all the functionality offered in our GUI and developed a list of the most frequently used applications, these are the manager and employee functions such as leave request or travel expenses.

It was from this idea that SAP Fiori was born.

We also decided to renovate business suite programs and enable customers to improve their user experience on their own.

For example, SAP developed SAP Screen Personas which allow customers to optimize and simplify any screen in the GUI. While developing SAP Fiori we also decided to renovate our business suite programs and provide enablement tools to enable customers to improve their user experience on their own, for example, SAP screen Personas that allow customers to optimize and simplify any screen in the GUI.

While working on renewing our existing solutions and enabling our customers we also worked on developing new applications to fulfill customer demands and needs.

© Copyright. All rights reserved.

Why Focus on User Experience?

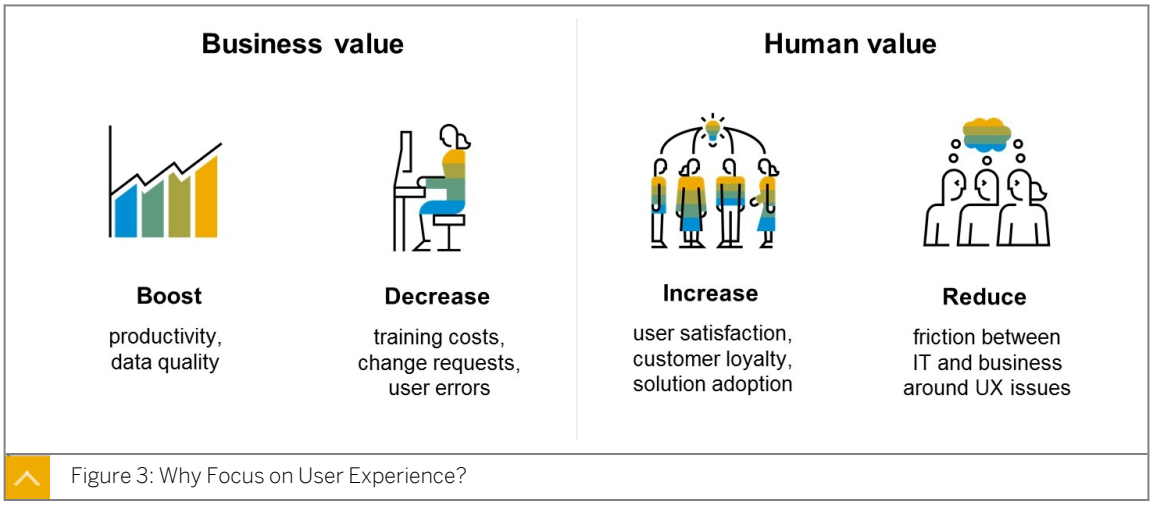


Figure 3: Why Focus on User Experience?

A good user experience is not just pretty screens. An intuitive UX delivers benefits you can quantify.

For example:

- A good user experience makes users more productive because they are able to move through business processes faster.
- An intuitive UX decreases training time (and costs).
- A simple, consistent user experience minimizes the risk of user errors leading to increased data quality and less rework.
- Looking beyond hard numbers, it's also likely you'll see happier users who are more likely to embrace and advocate for these new applications. You may also see reduced friction between IT and end-users, as screens should be easier to navigate and use.

The figure, Why Focus on User Experience?, shows the influences on monetary and human value.

© Copyright. All rights reserved.

SAP User Experience Design Services

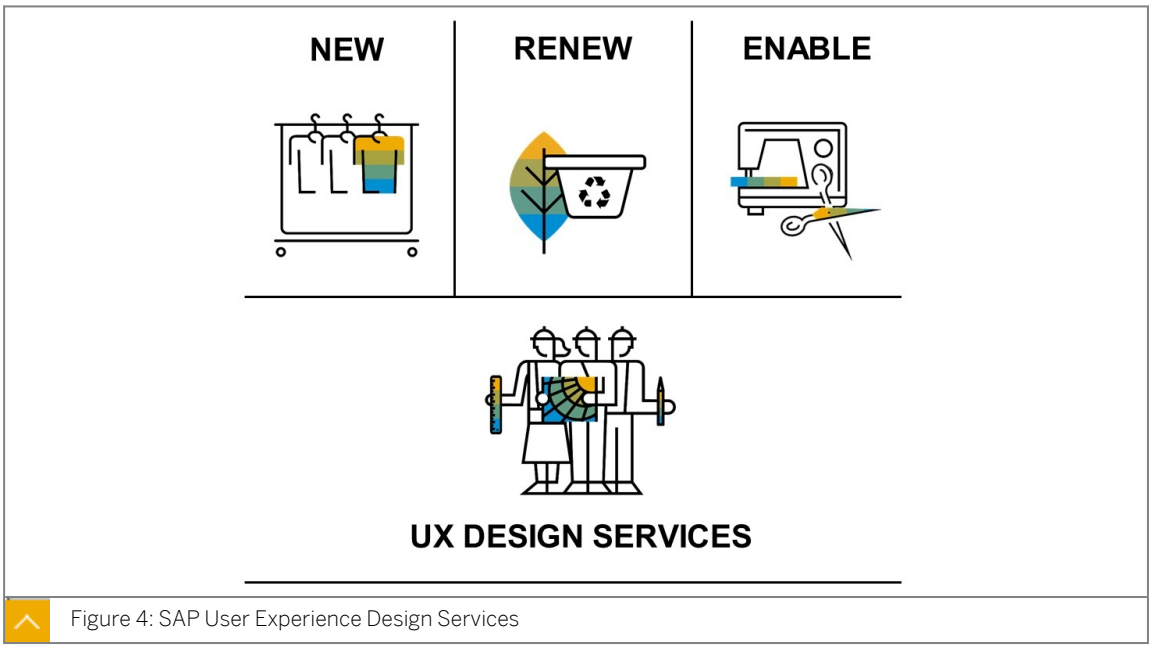


Figure 4: SAP User Experience Design Services

In the following sections, we look deeper into the User Experience (UX) Design Services component.

User Experience Strategy 2

Overview of SAP UX Design Services

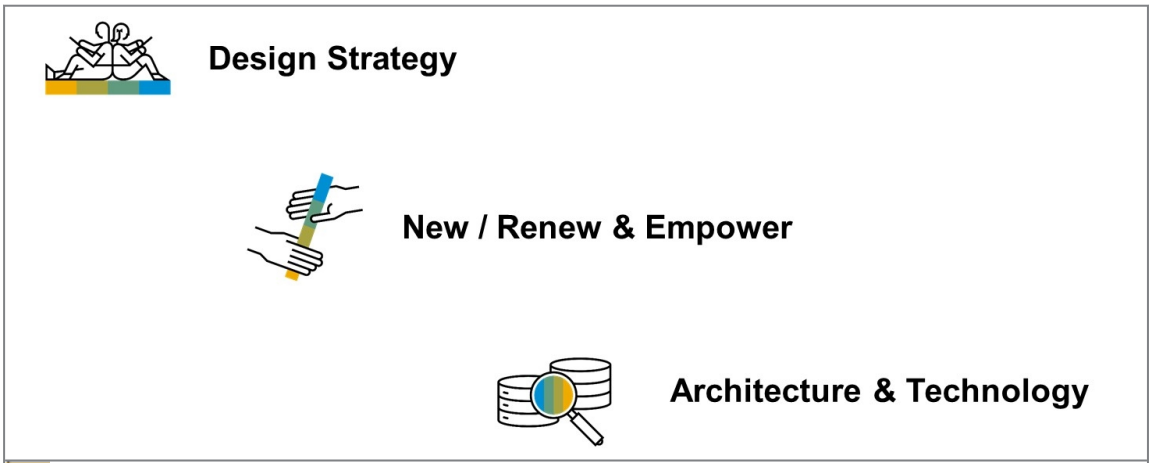


Figure 5: Overview of SAP UX Design Services

SAP Identified several leads from our customers and they asked us for advice on how to understand the SAP strategy and translate it to their reality. They asked for services to realize, implement, adapt and optimize the user experience of existing software.

If you achieved results such as with screen personas or SAP Fiori, the next level is to have customers build up skills on their own or empower their organization for a user experience strategy.

The last level is for the customer to become really innovative. This is the final goal of all customers, to be more innovative and they can achieve that by designing new products, looking for new services.

© Copyright. All rights reserved.

All of this is powered by design thinking.

Execution of the SAP UX Strategy

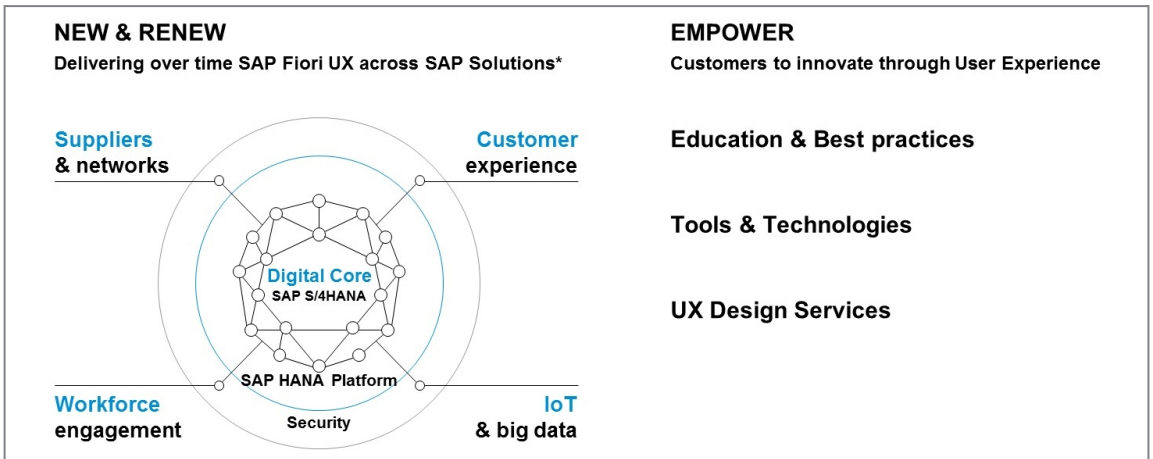


Figure 6: Execution of the SAP UX Strategy

The figure, Execution of the SAP UX Strategy, shows the execution of the SAP UX strategy.

Mindset

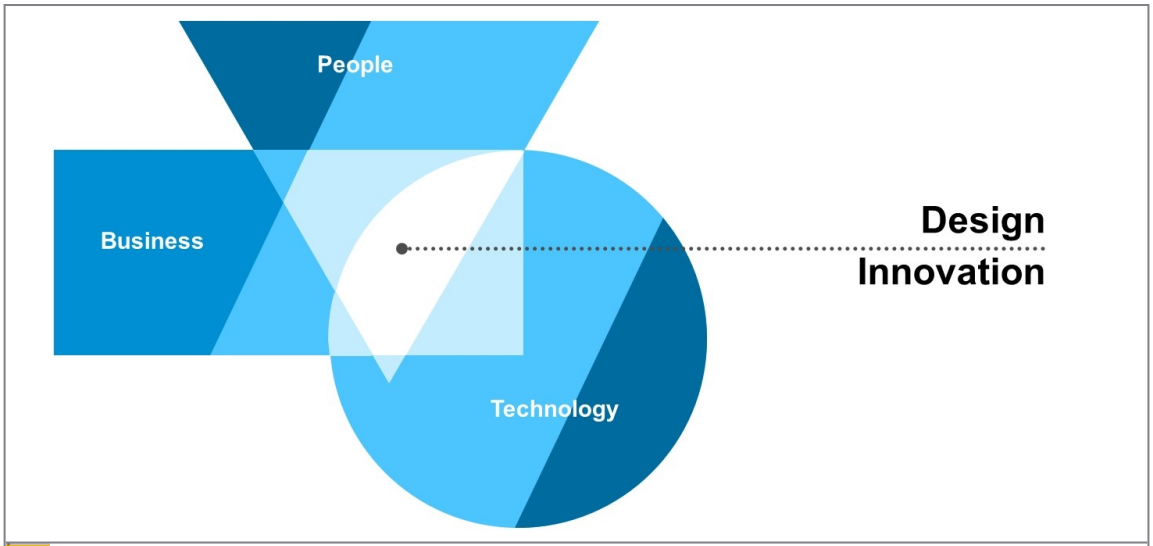


Figure 7: Mindset

SAP expanded their research to include the most commonly used functionality across all Lines of Business. With the first set of SAP Fiori apps we focused on the most commonly used business functions and those focused on the HR, workflow and SRM lines of business.

© Copyright. All rights reserved.

SAP UX Direction: SAP Fiori

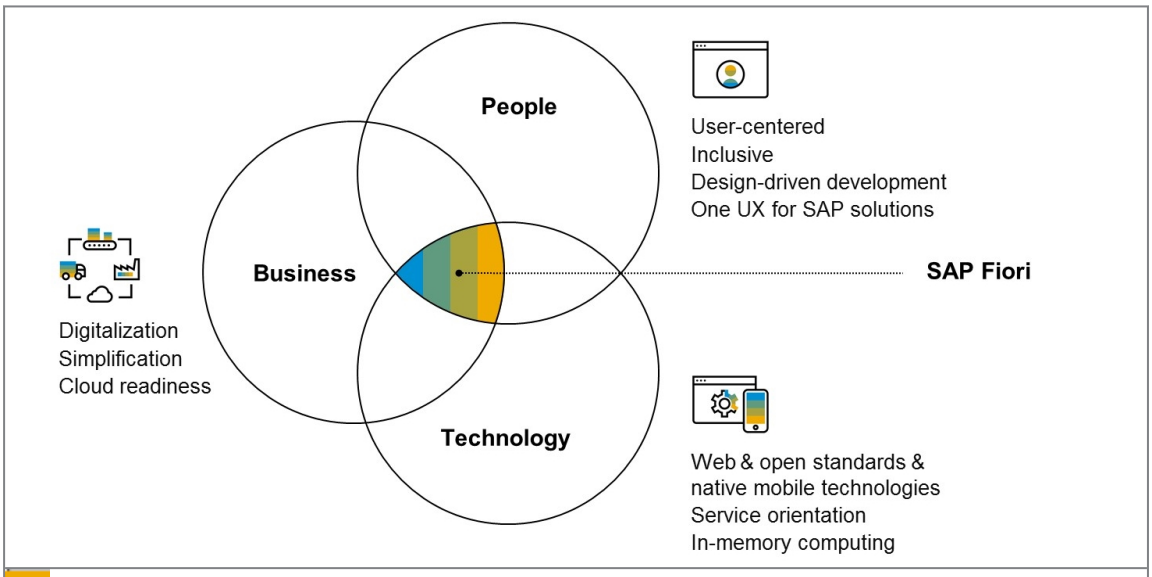


Figure 8: SAP UX Direction: SAP Fiori

SAP Fiori is the way SAP provides a great user experience.

SAP has always focused on supporting your business, that is, providing the features and functions you need to run your business processes, and SAP has always used innovative technologies for its products. What is relatively new is the focus on People; over the last ten years or so we have increasingly brought people, that is, the end users, into the equation. We have established a user-centered, design-driven development approach, where we have so-called design gates to ensure that the necessary end user research has been done, and that the application design follows the SAP Fiori design guidelines. User centricity means being inclusive and taking care of ALL users, including those with disabilities.

SAP Fiori is the sweet spot where business and technology come together to best support your users, helping them get their work done more easily.

© Copyright. All rights reserved.

User Experience Strategy 3

SAP Fiori - Keep Simple Things Simple

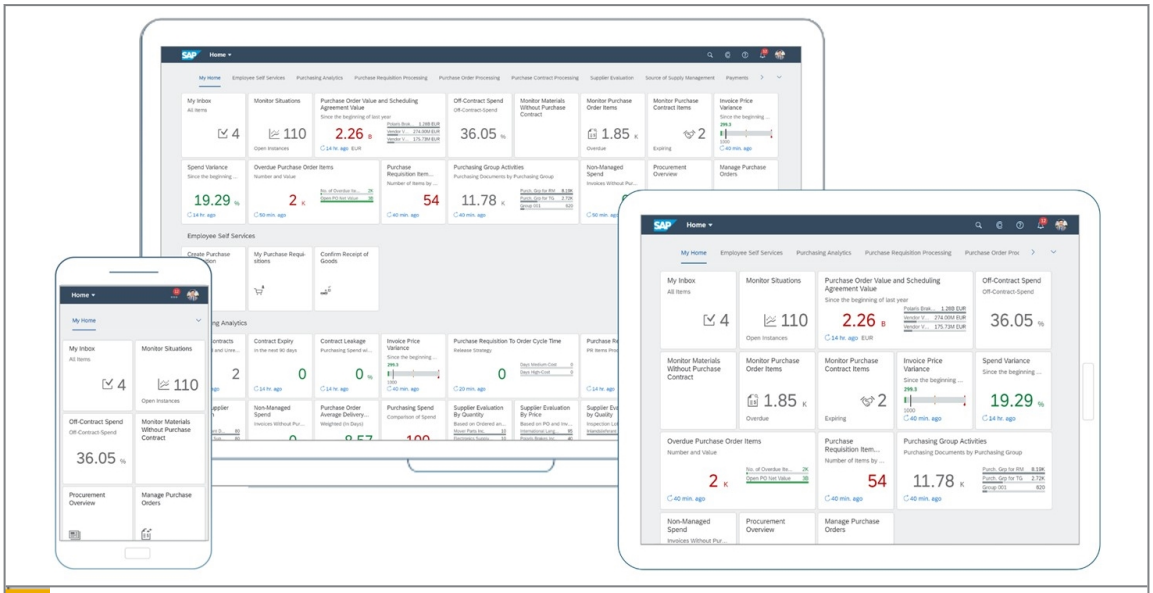


Figure 9: SAP Fiori - Keep Simple Things Simple

If you look at the tablet in this image you will see the detail list on the left side and the main pane. Now look to the mobile phone and notice only the main pane is visible. The detail list can be accessed by swiping the phone but both panes will not fit at once. Responsive design is credited for automatically completing the look on our UI framework.

An important thing to note with SAP Fiori is it can be deployed in the customers existing landscape. SAP customers running ECC 6.0 or Suite on Hana will need gateway to the back end system and add-ons making it an easy to deploy and use solution. An important thing to note with SAP Fiori is it can be deployed in the customers existing landscape. SAP customers running ECC 6.0 or Suite on Hana will need gateway to the back end system and add-ons making it an easy to deploy and use solution.

SAP Fiori Concept Review

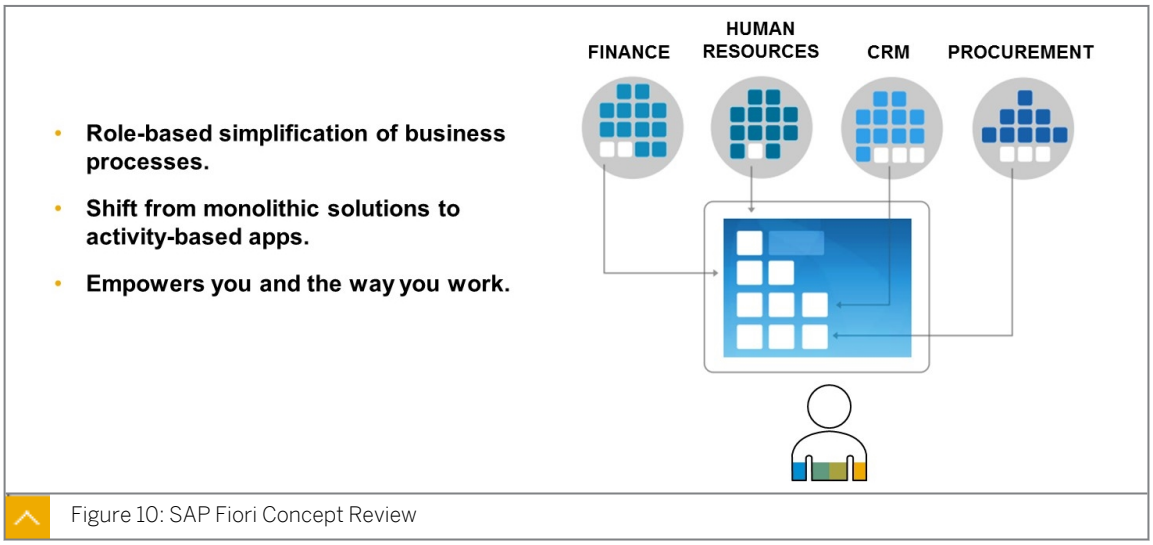


Figure 10: SAP Fiori Concept Review

© Copyright. All rights reserved.

The figure, SAP Fiori Concept Review, shows the entry to the SAP Fiori Concept.

Idea of SAP Fiori

- SAP Fiori is more than just a collection of apps, it represents the new SAP User Experience paradigm based on SAPUI5 framework
- Like SAP Fiori, SAPUI5 offers the developer opportunities to develop various business roles into simple, easy-to-use experiences for SAP software functions, and works seamlessly on desktop, tablet, or smartphone

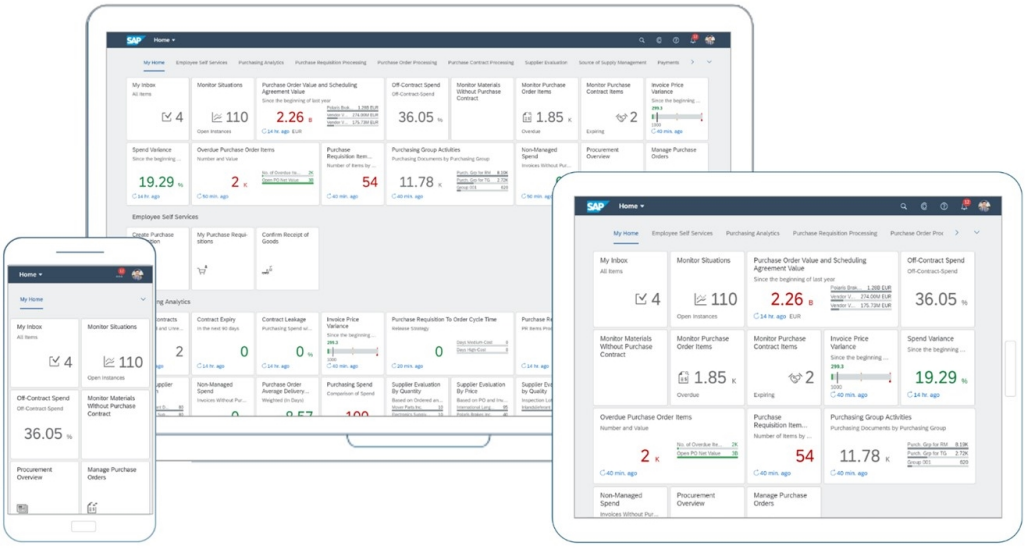


Figure 11: Idea of SAP Fiori

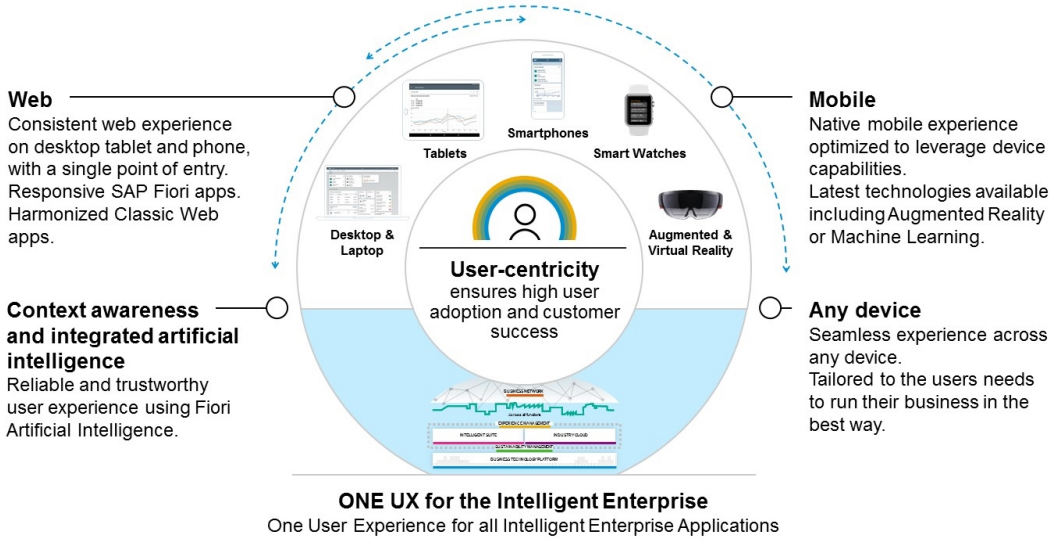


Figure 12: One User Experience is Key

The figure, Idea of SAP Fiori, shows examples of SAP Fiori on various devices.

© Copyright. All rights reserved.

SAP Fiori, Concept Review

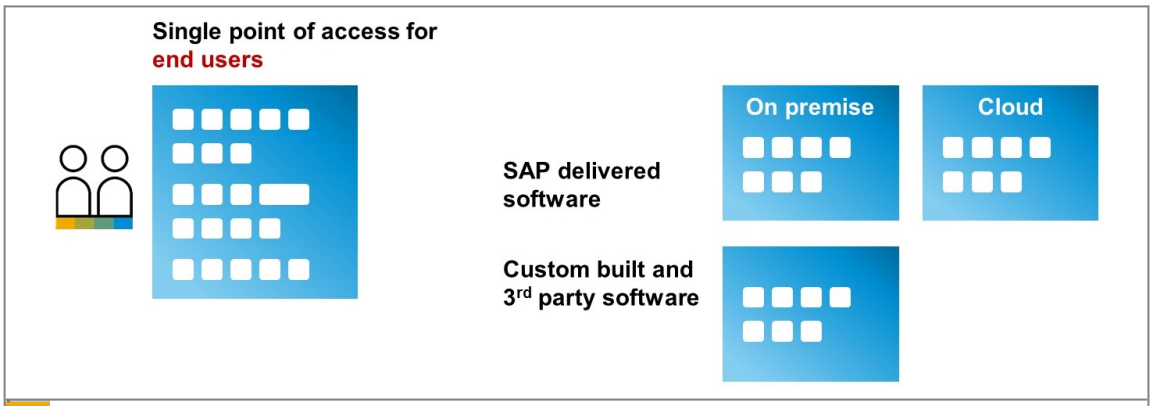


Figure 13: SAP Fiori, Concept Review

SAP Applications can be provided on-premise or as cloud applications. The user gets access to these applications using a single point of entry. This single point of entry is the SAP Fiori Launchpad (FLP).

SAP Fiori for SAP S/4HANA

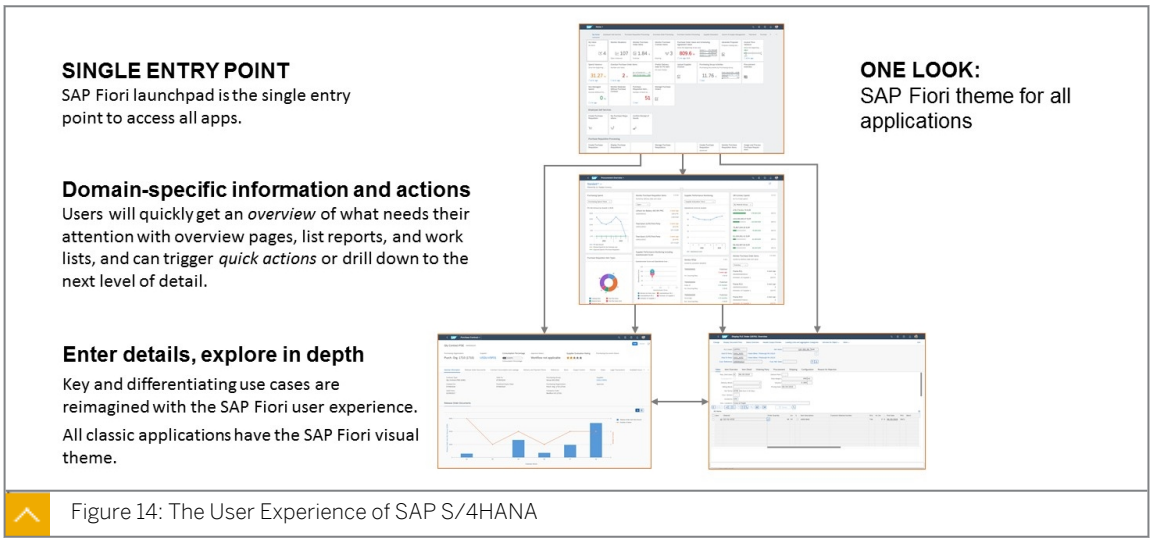


Figure 14: The User Experience of SAP S/4HANA

Level 1

Full role scope accessible via FLP, but mainly integration of legacy UIs.

For example: SAP Fiori Launchpad, Search, ...

Level 2

Domain specific info and action - All key business objects or domain areas.

Example: List Reports, Overview Pages, Work Lists, ...

Level 3

SAP Fiori UX for key use cases.

Example: Object Pages, Edit Pages, ...

Level 4

© Copyright. All rights reserved.

SAP Fiori Apps, WebGUI with SAP Screen Personas, WebDynpros, 3rd Party UIs, Partner UI.

Example: SAP Fiori apps for all use cases, Visually harmonized classic UIs

User Experience Strategy 4

UX as a Service

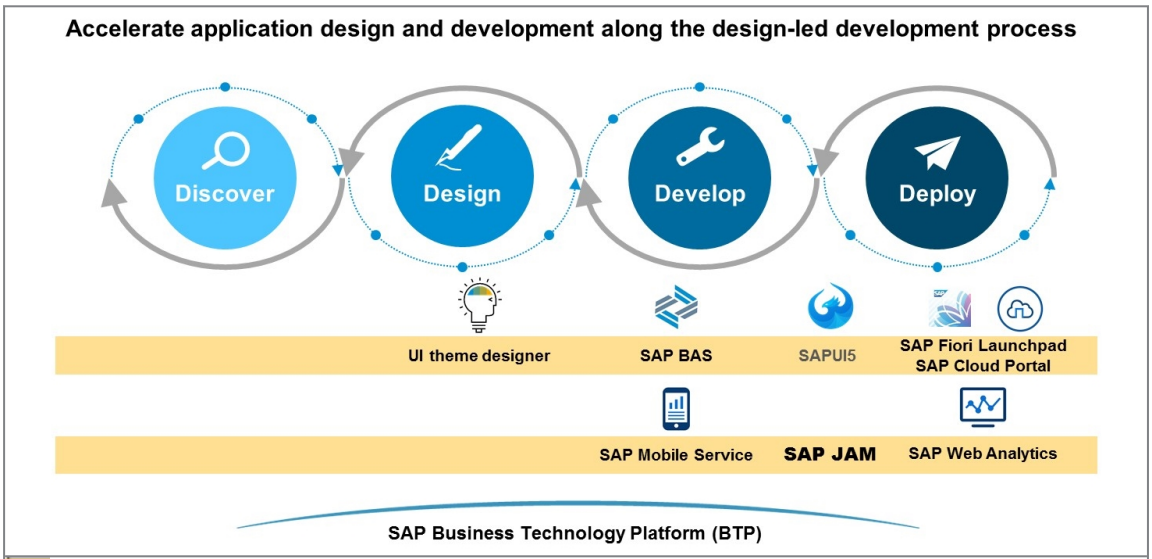


Figure 15: UX as a Service

SAP is committed to designing role-based applications that address the needs of our end users across all lines of business, tasks, and devices. We believe this is the key to a great user experience. But how do we guarantee a solid and consistent design for our customers and end users? The answer is SAP's design-led development process.

Design-led development takes advantage of proven design thinking methods to achieve an optimal user experience. The process spans the entire development lifecycle, is simple and easy to follow, and provides a solid basis for scaling design as a whole. It fosters unity between designers and developers, while ensuring that the needs of the end user are addressed at every step along the way. You can get more details under https://experience.sap.com/fiori-design-web/design-led-development-process-external/

The design-led development process is supported by various tools and technologies of SAP.

© Copyright. All rights reserved.

Summary of SAP UX Design Services

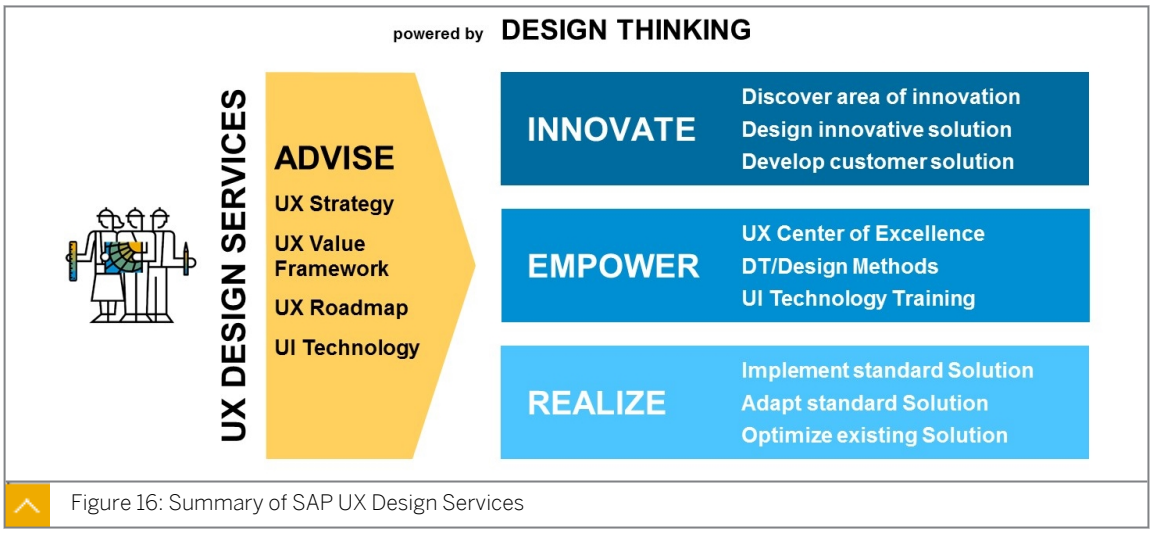


Figure 16: Summary of SAP UX Design Services

SAP has Identified several leads from our customers and they asked us for advice to understand the SAP strategy and translate it to their reality:

- Asked for services to realize: implement, adapt, and optimize the user experience of existing software.
- If you achieved results such as with screen personas or SAP Fiori, the next level is to have customers build up skills on their own or empower their organization for a user experience strategy.
- The last level is for the customer to become really innovative. This is the final goal of all customers, to be more innovative and they can achieve that by designing new products, looking for new services, and so on.
- All of this is powered by design thinking.
- In Summary: the design services that SAP offers are to advise about the strategy and the value of a solid user experience.

© Copyright. All rights reserved.

Renew User Experience of SAP Functions Across Lines of Business (LOB)

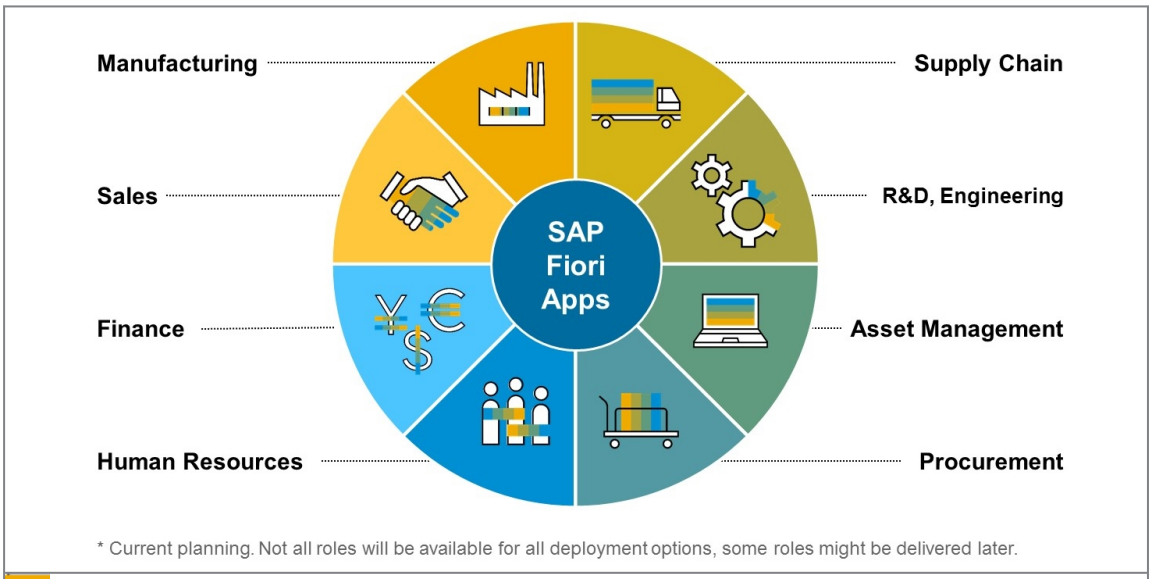


Figure 17: Renew User Experience of SAP Functions Across LOBs

The figure, Renew User Experience of SAP Functions Across LOBs, shows the various areas for which the renewed User Experience of SAP Functions Across LOBs is available.

SAP Fiori Architecture SAP S/4HANA Principle of One Archetype

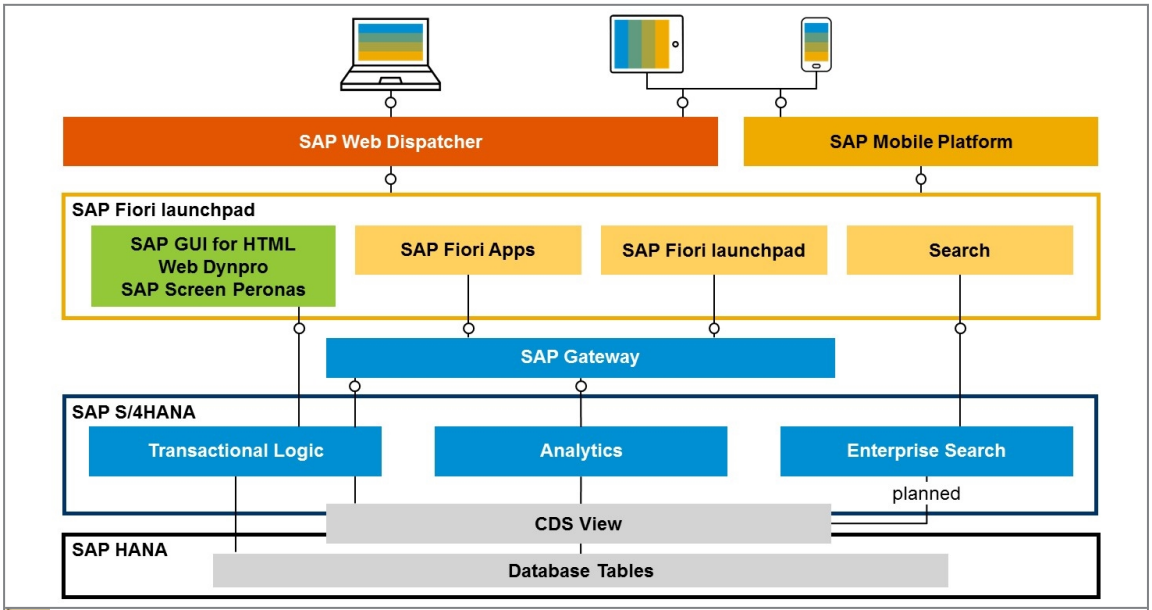


Figure 18: SAP Fiori Architecture SAP S/4HANA Principle of one Archetype

SAP Fiori Architecture for SAP S/4HANA consists of only one archetype for transactional, analytical, and search:

SAP Fiori technology components

- SAP Fiori launchpad

© Copyright. All rights reserved.

- Metadata driven UIs - Smart Controls & Smart Templates

ABAP infrastructure components

- Draft Infrastructure for transactional Logic
- SADL for CDS read access
- Analytical Engine (embedded BW) for analytical CDS access
- SAP Gateway for OData exposure

CDS Views (ABAP managed)

- Uniform Business Object Modeling
- Central repository for Metadata

Key Enablement Tools

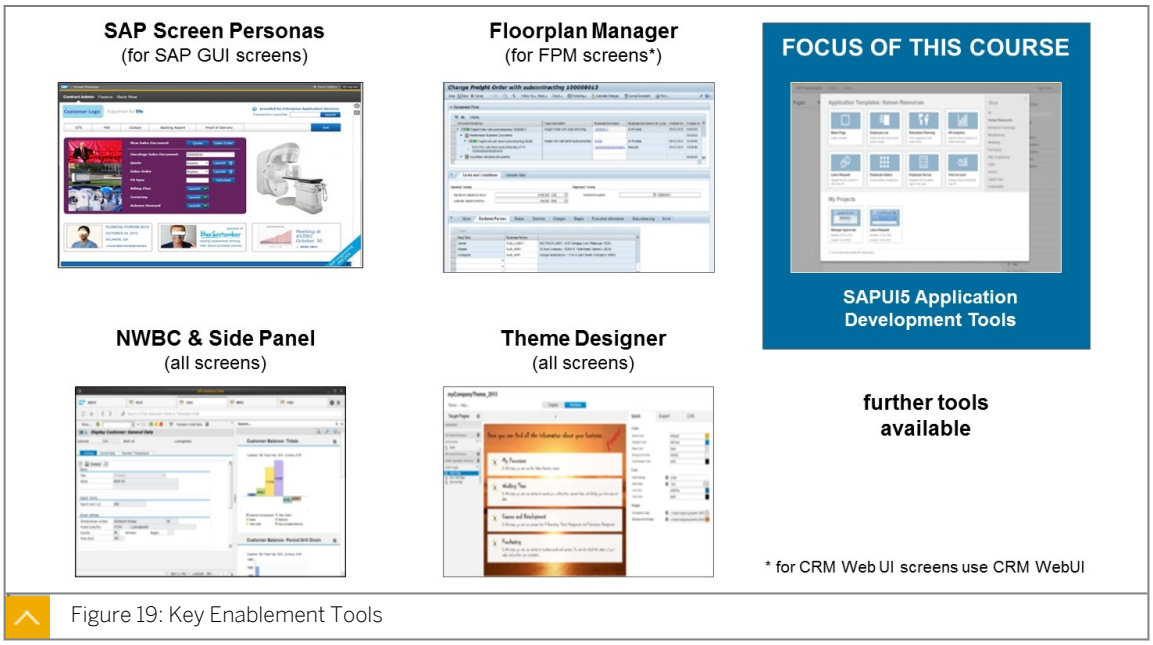


Figure 19: Key Enablement Tools

Floorplan Manager is based on ABAP Web Dynpro and allows customers to build new screens and adapt floorplan manager screens.

SAP UI5 application development tools are available so customers can build, or adapt SAP Fiori apps on their own or build their own UI5 applications.

NW business client, side panel and we offer a theme designer which is a tool that allows to adapt the branding of customer-specific branding (colors, fonts, logos, change style sheets, and so on).

LESSON SUMMARY

You should now be able to:

- Describe SAP User Experience Strategy

© Copyright. All rights reserved.

© Copyright. All rights reserved.

Explaining SAP User Experience Tools and Technologies

LESSON OBJECTIVES

After completing this lesson, you will be able to:

- Explain SAP User Experience Tools and Technologies

User Experience Tools and Technologies

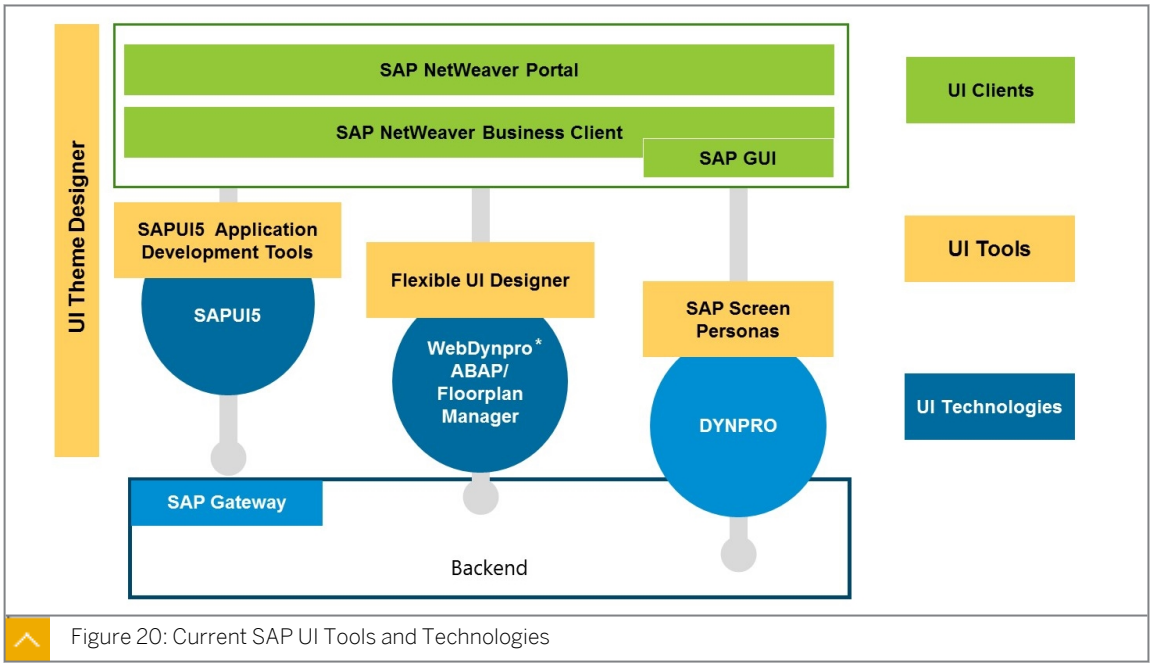


Figure 20: Current SAP UI Tools and Technologies

* Harmonized in Run & Design Time with WebUIF

SAP SAPUI5 and UI5 application development tools are used to change, adapt, or develop new applications.

Web Dynpro ABAP and Floorplan manager can be used for adoption or creating new apps.

SAP Dynpro includes SAP screen personas to optimize SAP GUI screens. In the figure, Current SAP UI Tools and Technologies, notice the connections to the back end - Dynpro screens have a close relationship to the back end, they are interwoven. SAP Screen Personas are only a layer on Dynpro where you can reduce fields, merge tops, and combine fields but you cannot add business functionality - this only works on the UI level.

With SAP UI5 and Gateway, there is a clear separation between UI and business logic. This setup enables you to be much more flexible in the future. Gateway does not need to run on the same machine as the back end.

© Copyright. All rights reserved.

Theme designer runs on the right side of the screen and allows for the branding of UIs.

A basic fundamental architectural change is that SAP decoupled UI and business logic which makes it easier for developers to react to changes in UI technology. SAP's UI technologies will continue to evolve and improve into new UI technologies as they make it to the marketplace in the future.

Features of the SAP SAPUI5 Framework

The following list outlines the features of the SAP SAPUI5 Framework:

- The UI development toolkit for HTML5 (also known as SAP SAPUI5) is a user interface technology that is used to build and adapt client applications.
- The SAPUI5 runtime is a client-side HTML5 rendering library with a rich set of standard and extension controls.
- It provides a lightweight programming model for desktop and mobile applications.
- Based on JavaScript, it supports the following RIA like client-side features:
  - SAPUI5 complies with OpenAjax and can be used together with standard JavaScript libraries.
- It supports CSS3, which allows you to adapt themes to your company's branding in an effective manner.
- It is based on an extensibility concept regarding custom controls and uses the open-source jQuery library as a foundation.

SAPUI5 is a UI technology that provides everything you need to build enterprise-ready web apps. It comes with all main SAP platforms but can also be used outside the SAP ecosystem because a large part of SAPUI5 had been open-sourced with OpenUI5.

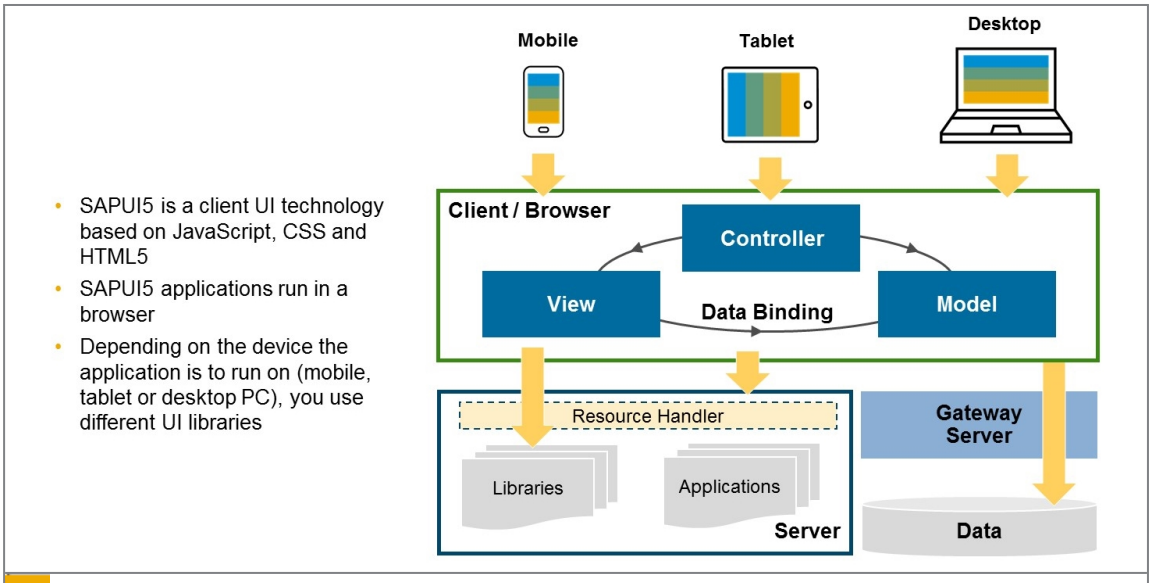


Figure 21: SAP SAPUI5 Architecture

Servers come into play for deploying your applications, storing the SAPUI5 libraries and connecting to a database. Depending on the environment in which SAPUI5 is used, the libraries or your applications are stored on an SAP NetWeaver Application Server or an SAP

© Copyright. All rights reserved.

HANA Cloud platform, for instance, the preferred way to access business data for your application is using the OData model through an SAP NetWeaver Gateway.

When users access an SAPUI5 application from their device, a request is sent to the respective server to load the application into the browser. The view accesses the relevant libraries. Usually the model is also instantiated and business data is fetched from the database.

SAPUI5 Provides Predefined Models

SAPUI5 provides predefined models. These are the benefits:

The JSON model can be used to bind controls to JavaScript object data, which is usually serialized in the JSON format.

- The JSON model is a client-side model and, therefore, intended for small datasets, which are available on the client.
- The JSON model supports two-way binding.

The XML model is a client-side model intended for small datasets, which are available on the client.

The XML Model does not contain mechanisms for server-based paging or loading of deltas.

The Resource model is designed to handle data in resource bundles, mainly to provide texts in different languages.

The OData model enables binding of controls to data from OData services

- The OData model is a server-side model: the dataset is only available on the server and the client only knows the currently visible rows and fields.
- This means that sorting and filtering on the client is not possible.
- For this, the client has to send a request to the server.
- The OData model supports OData version 2.0 and 4.0.

SAPUI5 supports different types of models. A model in the Model View Controller concept holds the data and provides methods to retrieve the data from the database and to set and update data.

LESSON SUMMARY

You should now be able to:

- Explain SAP User Experience Tools and Technologies

© Copyright. All rights reserved.

© Copyright. All rights reserved.

Describing SAP User Experience Use Case for Building Fiori-like Apps

LESSON OBJECTIVES

After completing this lesson, you will be able to:

- Describe SAP User Experience Use Case for Building Fiori-like Apps

SAP User Experience Use Case for Building Fiori-Like Apps

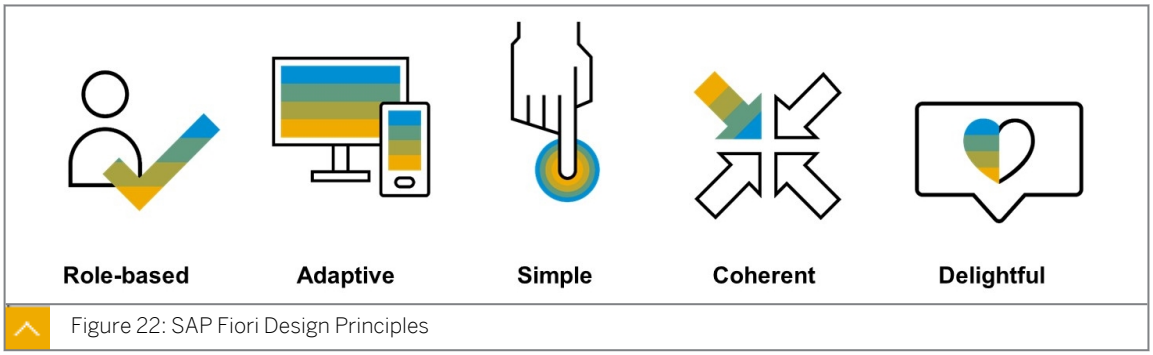


Figure 22: SAP Fiori Design Principles

The following list outlines the SAP Fiori Design Principles:

Role based:
Designed for you, your needs, and how you work.

Adaptive:
Adapts to multiple use cases and devices.

Simple:
Only what is necessary.

Coherent:
Provides one fluid user experience.

Delightful:
Makes an emotional connection.

Use Case - Design Process for Fiori-Like Apps Using SAPUI5

The following is the Design Process for Fiori-Like Apps using SAPUI5:

1. Observe how users are currently using the application to complete a business process.
2. Facilitate a design thinking workshop with users and designers.
3. Create wireframes to represent the agreed upon use case.

© Copyright. All rights reserved.

4. Validate the wireframes with the users to ensure that their use case was understood correctly.
5. Iterate through the wireframe scenarios and user validation until agreement is met.
6. Make proposal of finalized wireframes to all stakeholders.
7. Begin the development and test cycles of the SAPUI5 application.

Design-Led Development (DLD)

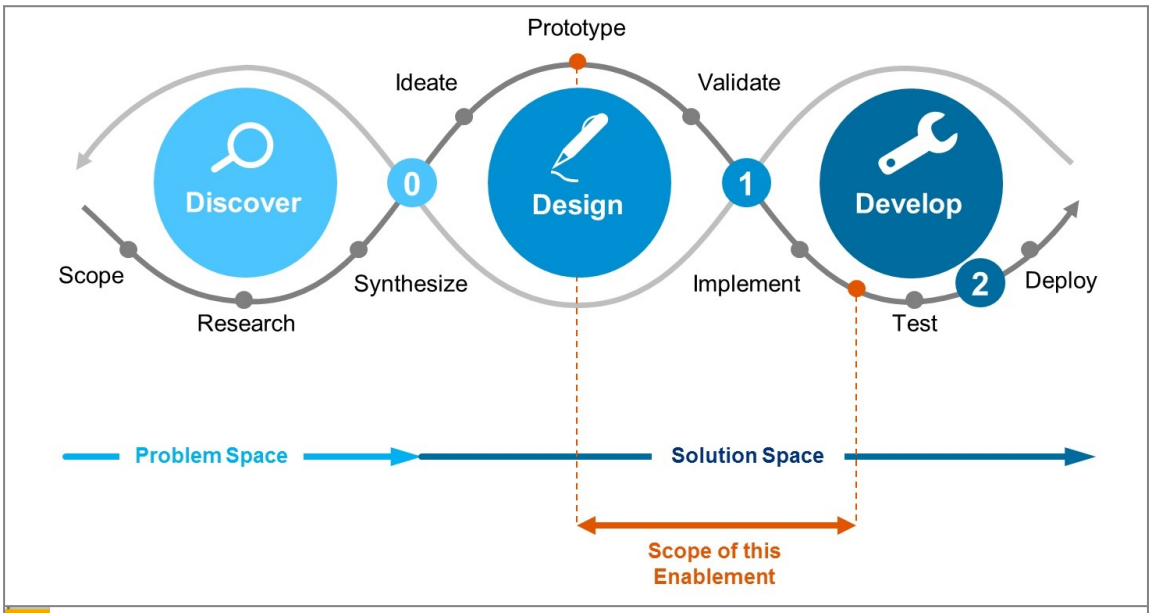


Figure 23: Design-Led Development (DLD)

How is UX achieved? It is achieved through an iterative approach in the early stages of software development.

Use Case - Financial Dashboard Outcome

Business Problem: The standard implementation of a financial dashboard was cumbersome and difficult to use: in the figure, Use Case - Financial Dashboard Outcome, see the *Before* bubble.

Solution: By using the user experience driven design process, a user-centric financial dashboard cockpit was developed that focused on the core four Key Performance Indicators (KPIs) across all users. see the *After* bubble.

© Copyright. All rights reserved.

Online SAP SAPUI5 Application Demonstrations

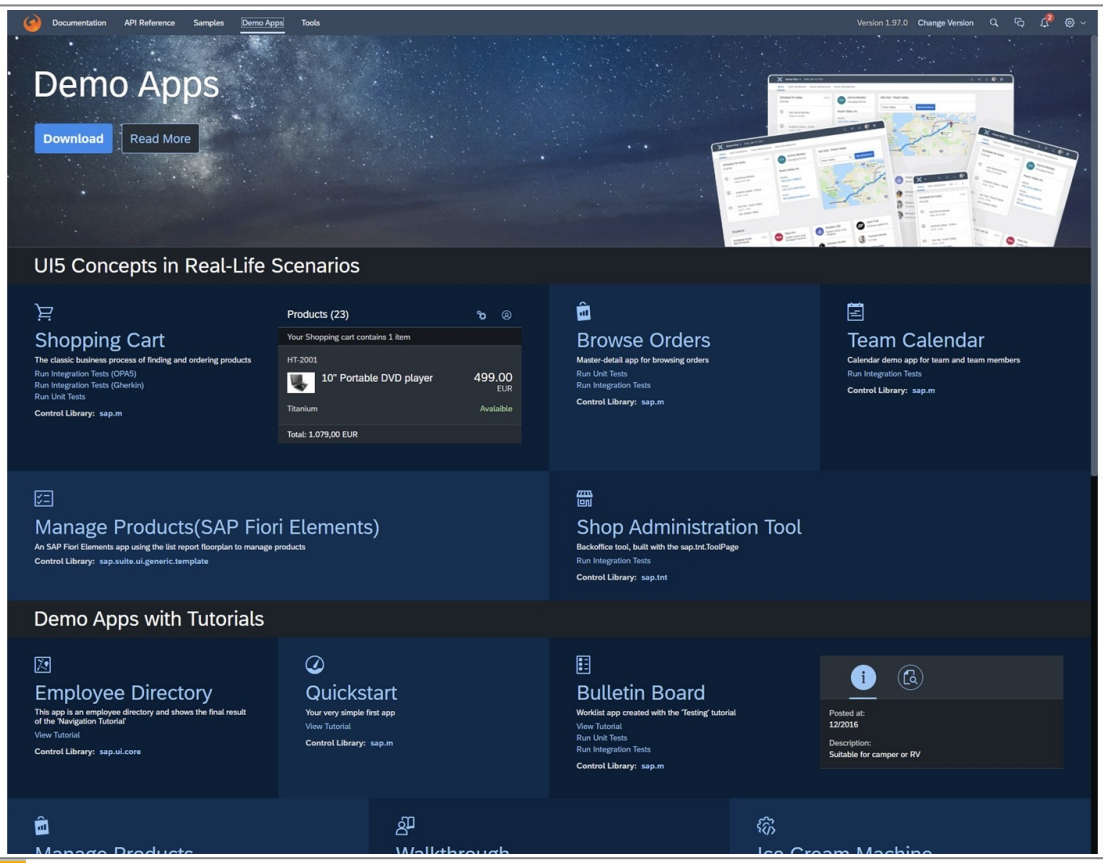


Figure 24: Online SAP SAPUI5 Application Demonstrations

The figure, Online SAP SAPUI5 Application Demonstrations, shows the demo entry page.

Take a moment to view the sample application that SAP has made available online at: https://sapui5.hana.ondemand.com/#/demoapps

SAP Business Application Studio

The following section provides an introduction to SAP Business Application Studio: Speeding Development by Using One Tool.

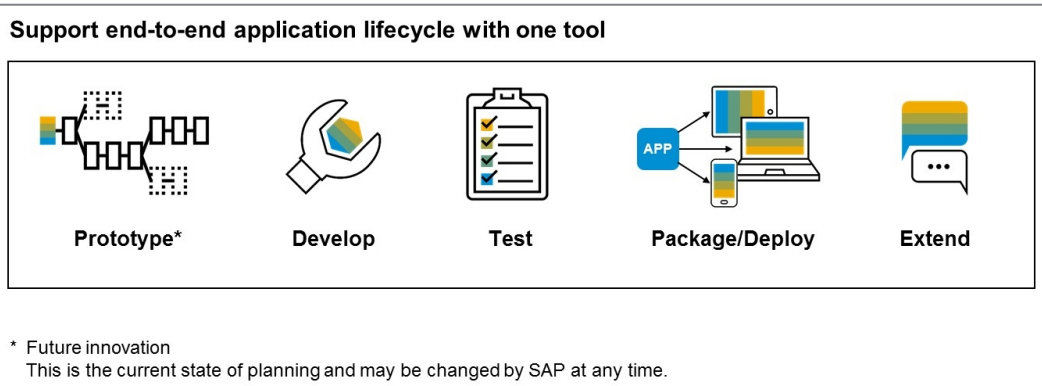


Figure 25: Introduction to SAP Business Application Studio: Speeding Development by Using One Tool

© Copyright. All rights reserved.

To implement SAP Fiori applications, it is recommended to use the SAP Business Application Studio. SAP Business Application Studio is a browser-based tool that empowers developers, business experts, and designers to build new user interfaces that work with SAP applications.

SAP Business Application Studio intends to simplify the end-to-end application lifecycle: prototyping, development, packaging, deployment, and customer extensions for SAPUI5 applications.

- Prototyping
- Developing
- Packaging
- Deploying
- Extending (customer extensions for SAPUI5 applications)
- Adapting

Introduction to SAP Business Application Studio

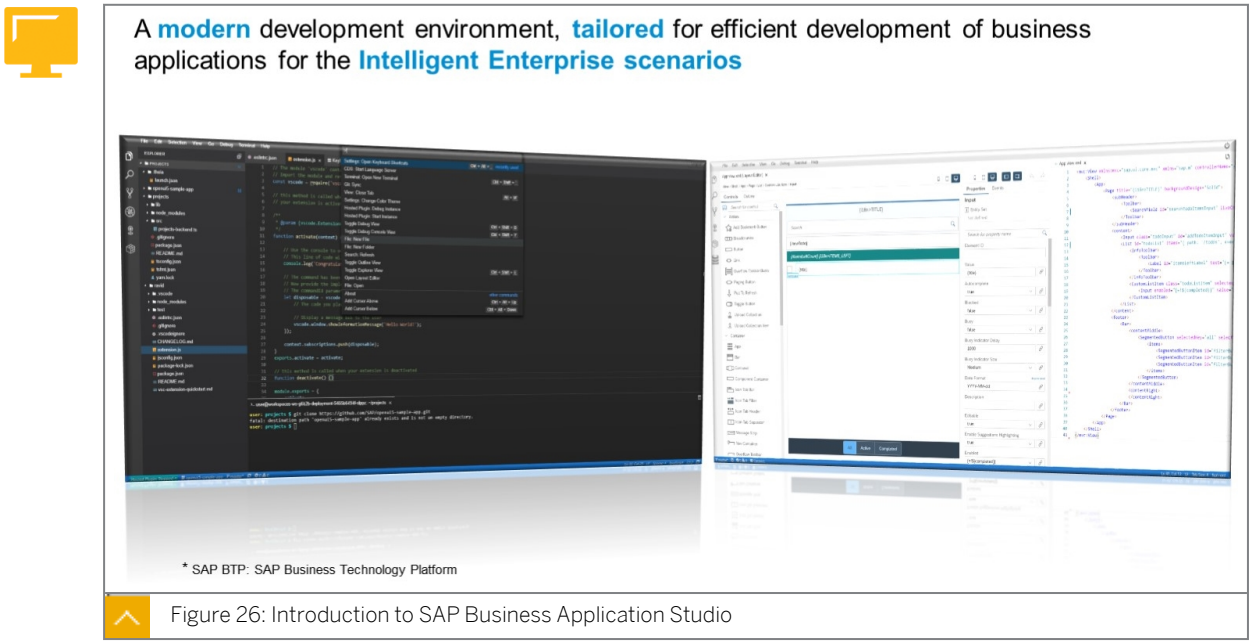
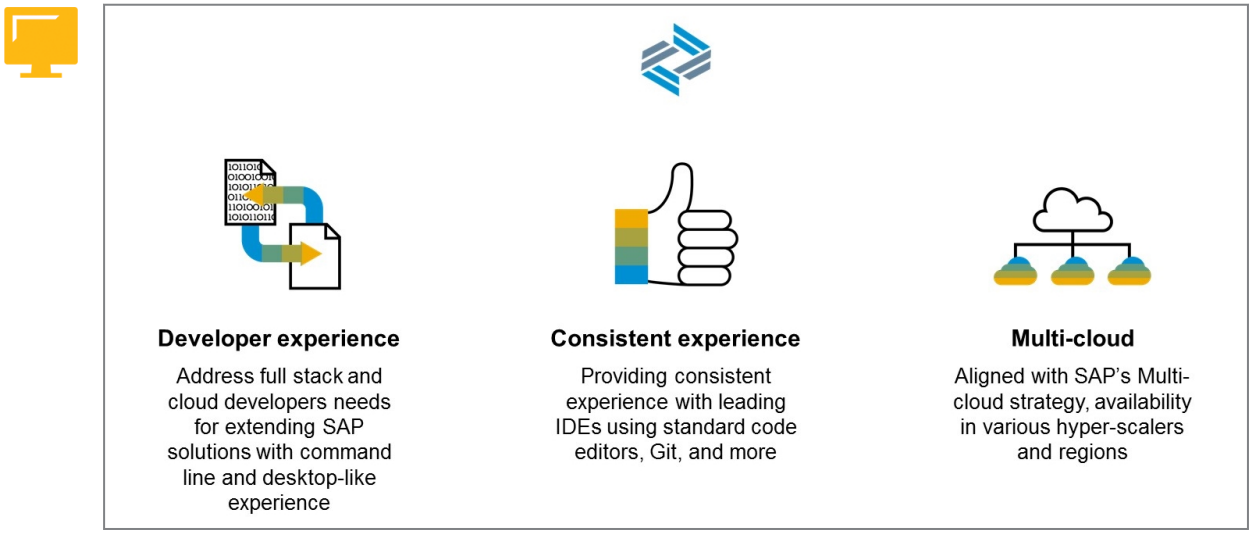


Figure 26: Introduction to SAP Business Application Studio



© Copyright. All rights reserved.

SAP Business Application Studio is the next generation of SAP Web IDE. It was made generally available on SAP's multi-cloud environment in Jan/2020 (Azur, AWS, and Ali-Cloud), and is also available on SAP BTP trial. It's a modern development environment, tailored for efficient development of business applications for the intelligent enterprise such as: SAP Fiori, SAP S/4 HANA extensions, modeling workflows for SAP BTP Workflow, and applications for SAP Cloud Mobile Services. The development efficiency is due to many high-productivity tools that are available out-of-the-box with SAP business Application Studio.

Introduction to SAP Business Application Studio II

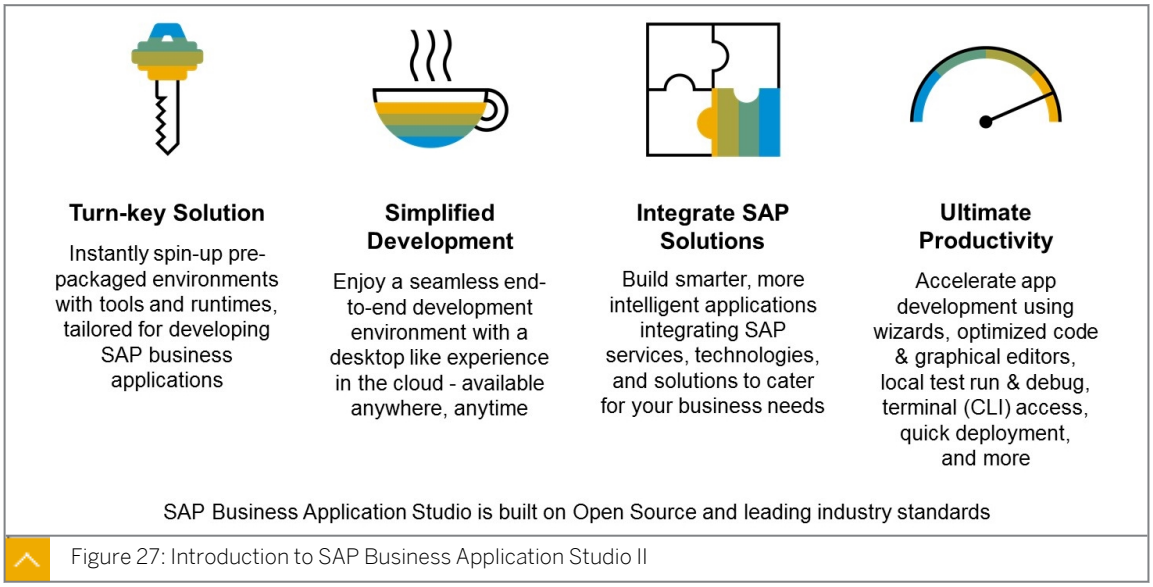


Figure 27: Introduction to SAP Business Application Studio II

- Turn-key solution - It provides turn-key solutions based on Dev-Spaces. A Dev-Space is a pre-configured environment (virtual machine on the cloud) where you can develop, build, test, and run using pre-installed runtimes and tools. Each Dev-space is suited for the development of the target scenarios such as SAP S/4HANA extensions, Business Services, Fiori, and more.
- Simplified Development - SAP Business Application Studio is based on standards and open sources, for example, Eclipse Theia IDE, which embraces VS Code extensions and experience.
- Integrate SAP solutions – Allows easy connectivity to SAP systems to consume SAP data, for example, using API Hub. Quick integration to SAP solutions, for example, SAP S/4HANA.
- Ultimate Productivity - E2E development - onboarding, developing, debugging, testing, packaging, and deploying. Desktop-like development experience in the cloud (CLI, local tools....) , productivity tools, mock servers, and much more.

© Copyright. All rights reserved.

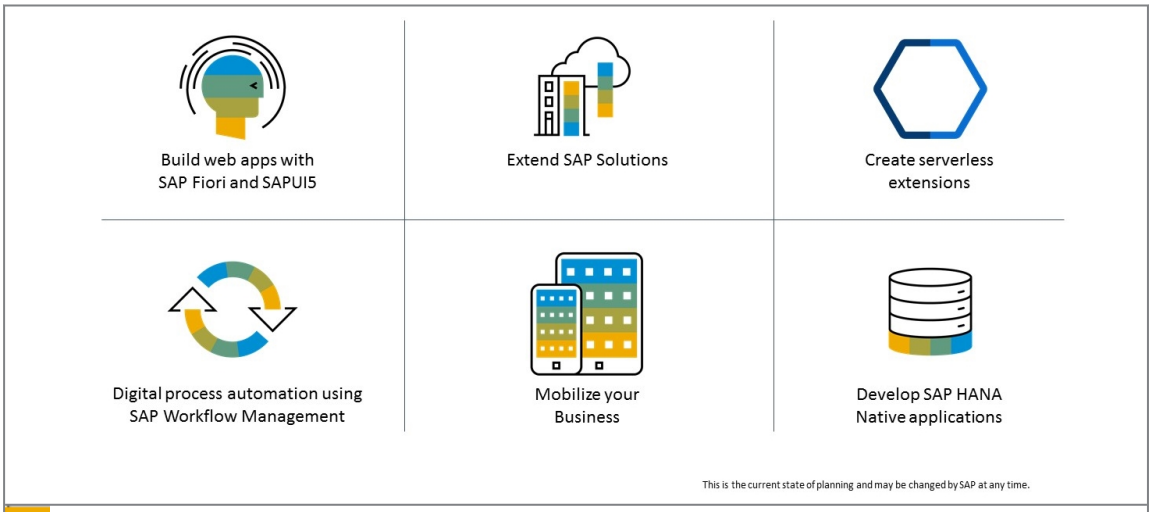
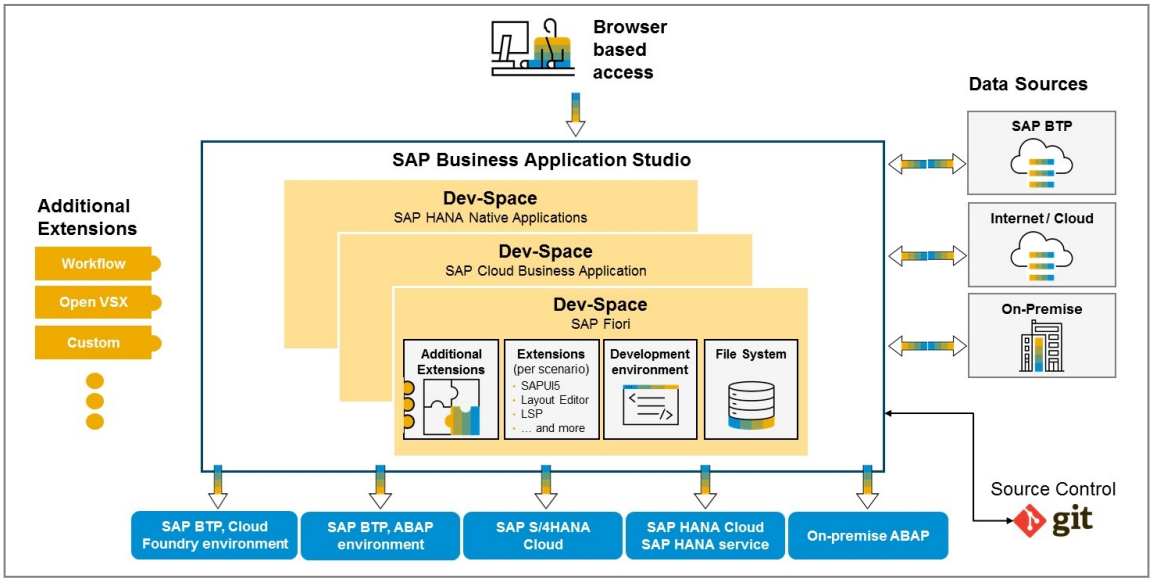


Figure 28: BAS Key Use Cases

- Create and extend SAP Fiori/SAPUI5 apps.
- Leverage out of the box code templates based on SAP best practices.
- Extend existing SAP Fiori applications via predefined extension points.
- Adapt the user interface of SAP Fiori elements applications.
- Use text editors and graphical tools.
- Easily deploy to various SAP platforms.
- Develop and extend SaaS and S/4HANA.
- End-to-end, single integrated development experience for all of the application's modules.
- Create an SAP HANA data model.
- Develop Java-based or node.js-based business logic.
- Consume the service with SAP Fiori UX.
- Model and deploy SAP BPM Workflows.
- Use template to create workflow module.
- Model workflows and forms using the workflow and form editors.
- Build the deployment artifact.
- Deploy directly from SAP Business Application Studio to SAP BTP, Cloud Foundry environment.
- Develop mobile applications
- High-control, create, test, build, and deploy Apache Cordova hybrid mobile apps.
- Codeless, metadata-driven application development
- Leverage mobile services integration.
- Create serverless extensions
- Tools to create and deploy project.

© Copyright. All rights reserved.

- Integrated development experience with other SAP services.
- Accelerated development with help of specialized wizards and code editors for serverless extensions with functions.
- Events exploration.
- Develop SAP HANA Native applications
- Create SAP HANA application project or add SAP HANA db module to existing project.
- Support various SAP HANA artifacts, such as: SQL editor, calculation view graphical editor, flow graph graphical editor, and database explorer.
- Deploy SAP HANA artifacts.



How does it work – A sneak pick under the hood of SAP business Application Studio:

The development environment is accessed via browser. This means that it's a hassle free environment. There is no need to install any special software or extension.

The developer can have multiple dev-spaces, which are like virtual machines in the cloud. This provides the desktop-like experience, with fast response time and more control over the environment.

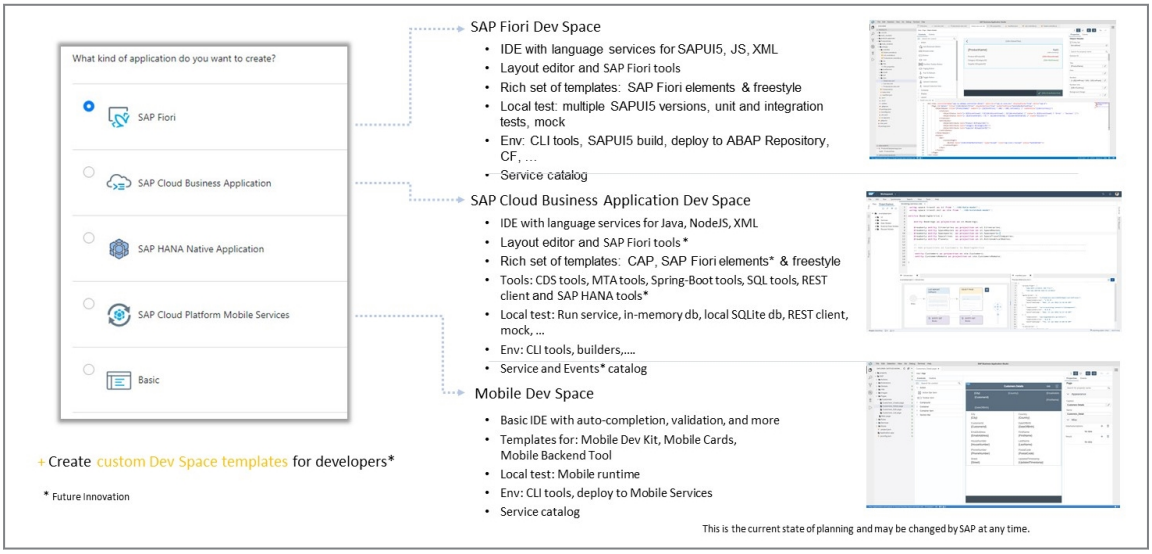
Each dev-space has pre-installed runtimes and tools needed for the use-case. This simplifies and saves time in setting up the development environment and allows to efficiently develop, test, build, and run the developed solution.

In this example, for SAP Fiori development, the dev-space has a file system where the code is stored (that we recommend as best practice to connect to a git source code repository). The development environment is an IDE based on an open source. It has extensions for SAP Fiori development, such as: Layout Editor, Language services for SAPUI5 (code completion, intellisense, and so on). If additional tools are needed, there's the option to customize the dev-space and add additional extensions.

During development, and later when testing in the actual runtime environment, the app can consume data and services coming from various sources: on-prem ABAP, SAP BTP, as well as non-SAP.

© Copyright. All rights reserved.

Once you are done developing, you can either test your code locally on a simulated runtime or easily deploy your app to the cloud and test it in the actual runtime environment on Cloud Foundry or on-prem ABAP.



The dev-spaces are at the heart of SAP Business Application Studio. Let's take a closer look at this concept:

- There are various dev-space types. Each dev-space type fits an Intelligent Enterprise development use case
- This slide presents additional examples of dev-space types. Each of these dev-space types comes prepackaged with the tools and runtimes relevant for its scenario. If we take for example the SAP Fiori dev space

  SAP Fiori:

- Open source IDE with editors & language services
- SAP Fiori templates
- Local test tools
- SAPUI5 build and the ability to deploy to the actual target runtime directly from SAP AppStudio

  In the future there will be the option to customize the dev-space as well as share, so all developers in the team will use the same set of development tools.

© Copyright. All rights reserved.

Introduction to SAP Business Application Studio: SAP BTP Connectivity

SAP BTP Connectivity allows SAP BTP applications to securely access remote services that run on the Internet or on-premise.

This component:

- Allows subaccount-specific configuration of application connections via destinations.
- Provides a Java API that application developers can use to consume remote services.
- Allows you to make connections to on-premise systems, using the Cloud Connector.
- Lets you establish a secure tunnel from your on-premise network to applications on SAP BTP, while you keep full control and auditability of what is exposed to the cloud.
- Supports both the Neo and the Cloud Foundry environment for application development on SAP BTP.

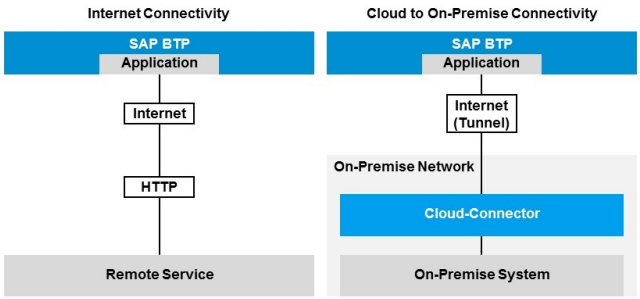


Figure 29: Introduction to SAP Business Application Studio: SAP BTP Connectivity

If you want to connect to an ABAP system, you need to specify this in the destination details.

Let's suppose you want to connect to the GM6 ABAP system (https://wdflbmt2291:55080) in order to consume some OData for creating a new SAP Fiori app or for extending some existing ones.

Furthermore, you want to execute some other SAPUI5 applications. We need to pass this information to the SAP Business Application Studio through the additional property WebIDEUsage of the destination. We need to specify also the following three additional properties:

- HTML5.DynamicDestination
- HTML5.Timeout
- WebIDEEnabled

Please refer to the official SAP Business Application Studio documentation to get more information on this.

LESSON SUMMARY

You should now be able to:

- Describe SAP User Experience Use Case for Building Fiori-like Apps

© Copyright. All rights reserved.

© Copyright. All rights reserved.

Learning Assessment

1. What impact does UX have on monetary values?

   Choose the correct answers.

   - ☐ **A** Increase user satisfaction
   - ☐ **B** Gain productivity and data quality
   - ☐ **C** Strengthen relationship between customers
   - ☐ **D** Save training costs
   - ☐ **E** Decrease change requests and user errors

2. What are the ideas behind the SAP UX strategy?

   Choose the correct answers.

   - ☐ **A** Design Strategy
   - ☐ **B** New / Renew / Enablement
   - ☐ **C** New / Renew / Empower
   - ☐ **D** Architecture and Technology
   - ☐ **E** SAP Screen Personas

3. What impact does SAP Fiori have on Business?

   Choose the correct answers.

   - ☐ **A** Digitalization
   - ☐ **B** Simplification
   - ☐ **C** Web & open standards
   - ☐ **D** User-centered
   - ☐ **E** Re-imagine processes

© Copyright. All rights reserved.

4. What are the current SAP UI Tools?

   Choose the correct answers.

   - ☐ A SAPUI5 Application Development Tools
   - ☐ B SAP Screen Personas
   - ☐ C SAP NetWeaver Portal
   - ☐ D Flexible UI Designer
   - ☐ E SAP NetWeaver Gateway

5. What are the current UI Technologies of SAP?

   Choose the correct answers.

   - ☐ A Business Server Pages
   - ☐ B SAPUI5
   - ☐ C Java Server Pages
   - ☐ D Web Dynpro ABAP / Floorplan Manager
   - ☐ E Dynpro

6. What goals does the SAPUI5 framework have?

   Choose the correct answers.

   - ☐ A User Interface technology for building and adapting client applications
   - ☐ B User Interface technology for building and adapting server based applications
   - ☐ C Providing a lightweight programming model for desktop only applications
   - ☐ D Providing an extensible framework for building desktop and mobile applications.

7. What are the SAP Fiori principles?

   Choose the correct answers.

   - ☐ A Rolebased
   - ☐ B Adaptive
   - ☐ C Creative
   - ☐ D Coherent
   - ☐ E Complex

© Copyright. All rights reserved.

8. What steps are part of the discover phase in the DLD?

   Choose the correct answers.

   - ☐ **A** Scope
   - ☐ **B** Test
   - ☐ **C** Implement
   - ☐ **D** Research
   - ☐ **E** Synthesize

9. What steps are part of the design phase in the DLD?

   Choose the correct answers.

   - ☐ **A** Test
   - ☐ **B** Validate
   - ☐ **C** Prototype
   - ☐ **D** Scope
   - ☐ **E** Ideate

© Copyright. All rights reserved.

Learning Assessment - Answers

1. What impact does UX have on monetary values?

   Choose the correct answers.

   - [ ] A Increase user satisfaction
   - [X] B Gain productivity and data quality
   - [ ] C Strengthen relationship between customers
   - [X] D Save training costs
   - [X] E Decrease change requests and user errors

   Correct. It gains productivity and data quality, saves training costs, and decreases change requests and user errors.

2. What are the ideas behind the SAP UX strategy?

   Choose the correct answers.

   - [X] A Design Strategy
   - [ ] B New / Renew / Enablement
   - [X] C New / Renew / Empower
   - [X] D Architecture and Technology
   - [ ] E SAP Screen Personas

   Correct. Design strategy, new / renew / empower, and architecture & technology are the ideas behind SAP UX strategy.

© Copyright. All rights reserved.

3. What impact does SAP Fiori have on Business?

   Choose the correct answers.

   - [X] A Digitalization
   - [X] B Simplification
   - [ ] C Web & open standards
   - [ ] D User-centered
   - [X] E Re-imagine processes

   Correct. Digitalization, simplification, and re-imagine processes are the impacts of SAP Fiori.

4. What are the current SAP UI Tools?

   Choose the correct answers.

   - [X] A SAPUI5 Application Development Tools
   - [X] B SAP Screen Personas
   - [ ] C SAP NetWeaver Portal
   - [X] D Flexible UI Designer
   - [ ] E SAP NetWeaver Gateway

   Correct. SAPUI5 Application Development Tools, SAP Screen Personas, and Flexible UI Designer are the current SAP UI tools.

5. What are the current UI Technologies of SAP?

   Choose the correct answers.

   - [ ] A Business Server Pages
   - [X] B SAPUI5
   - [ ] C Java Server Pages
   - [X] D Web Dynpro ABAP / Floorplan Manager
   - [X] E Dynpro

   Correct. SAPUI5, Web Dynpro ABAP / Floorplan Manager, and Dynpro are the current UI Technologies of SAP.

© Copyright. All rights reserved.

6. What goals does the SAPUI5 framework have?

   Choose the correct answers.

   [X] A User Interface technology for building and adapting client applications

   [ ] B User Interface technology for building and adapting server based applications

   [ ] C Providing a lightweight programming model for desktop only applications

   [X] D Providing an extensible framework for building desktop and mobile applications.

   Correct. User Interface technology for building and adapting client applications and providing an extensible framework for building desktop and mobile applications are the goals of SAPUI5 Framework.

7. What are the SAP Fiori principles?

   Choose the correct answers.

   [X] A Rolebased

   [X] B Adaptive

   [ ] C Creative

   [X] D Coherent

   [ ] E Complex

   Correct. Rolebased, adaptive, and coherent are the SAP Fiori principles.

8. What steps are part of the discover phase in the DLD?

   Choose the correct answers.

   [X] A Scope

   [ ] B Test

   [ ] C Implement

   [X] D Research

   [X] E Synthesize

   Correct. Scope, research, and synthesize are the steps of the discover phase in the DLD.

© Copyright. All rights reserved.

9. What steps are part of the design phase in the DLD?

   Choose the correct answers.

   - [ ] A Test
   - [X] B Validate
   - [X] C Prototype
   - [ ] D Scope
   - [X] E Ideate

   Correct. Validate, prototype, and ideate are the steps of the design phase in the DLD.

© Copyright. All rights reserved.

© Copyright. All rights reserved.

UNIT 2 SAP Fiori Elements, Overview

UNIT OBJECTIVES

- Explain the concept of Fiori Elements
- Explain the concept of annotations of OData services
- Explain the concept of Smart Control
- Explain Templates for Fiori Elements

© Copyright. All rights reserved.

- Explore the Development Environment
- Explore the basic process of building Fiori Elements applications
- Use the CDS View and SADL
- Use SADL
- Explain Metadata Extension
- Learn about scenarios of Fiori Elements implementation

© Copyright. All rights reserved.

Unit 2
Lesson 1

Explaining the Architecture of Fiori Elements

LESSON OBJECTIVES

After completing this lesson, you will be able to:

- Explain the concept of Fiori Elements
- Explain the concept of annotations of OData services
- Explain the concept of Smart Control

SAP Fiori Elements

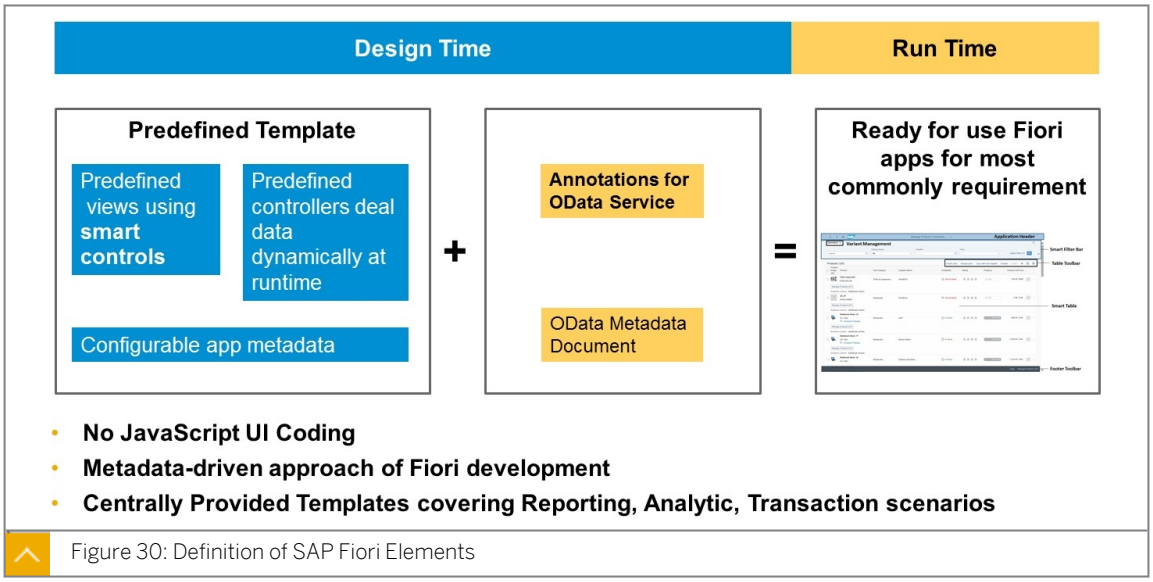


Figure 30: Definition of SAP Fiori Elements

SAP Fiori elements provide designs for UI patterns and predefined templates for commonly used application patterns. App developers can use SAP Fiori elements to create SAP Fiori applications based on OData services and annotations requiring no JavaScript UI coding. The resulting app uses predefined views and controllers that are provided centrally, so no application-specific view instances are required. The SAPUI5 runtime interprets metadata and annotations of the underlying OData service and uses the corresponding views for the SAP Fiori app at startup.

The predefined views and controllers ensure UI design consistency across similar apps. Also the metadata-driven development model significantly reduces the amount of front end code for each app, so the developer can focus on the business logic.

SAP Fiori Elements — Definition

No JavaScript UI Coding.

© Copyright. All rights reserved.

Not like traditional SAPUI5 development, detailed JavaScript knowledge is not mandatory for developing SAP Fiori elements applications. Since JavaScript is a flexible and totally dynamic language and no compile time code check, code written by JavaScript has more chance of runtime error. SAP Fiori elements will save you lots of effort of studying, writing and debugging JavaScript code. This approach of developing SAP Fiori apps can also your overall quality, stability, and maintainability of your SAP Fiori apps, especially when you need lots of SAP Fiori apps.

Metadata-driven approach of SAP Fiori development.

As you have learned from previous course, SAP Fiori applications rely on OData service, which provide back end logic. A metadata document of each OData service describes information of the service including entities, properties for each entity, collections, association and all stuffs for the service consumer. The approach of SAP Fiori elements development is to add more vivid descriptions to the OData metadata document. Those descriptions let the consumer not only know information of data, but also know how to represent the data. We call the descriptions added to OData metadata document as OData Annotation.

Centrally Provided Templates covering Reporting, Analytic, Transaction scenarios.

SAP is responsible for providing templates for SAP Fiori elements. Each template targets one common requirement of enterprise application. Currently, we have List Report, Overview Page, and Analytic List Page, targeting to daily reporting, analytic and transaction scenarios. SAP will continually add more templates to ensure SAP Fiori elements covers most of common requirements.

Concept of Annotations of OData Services

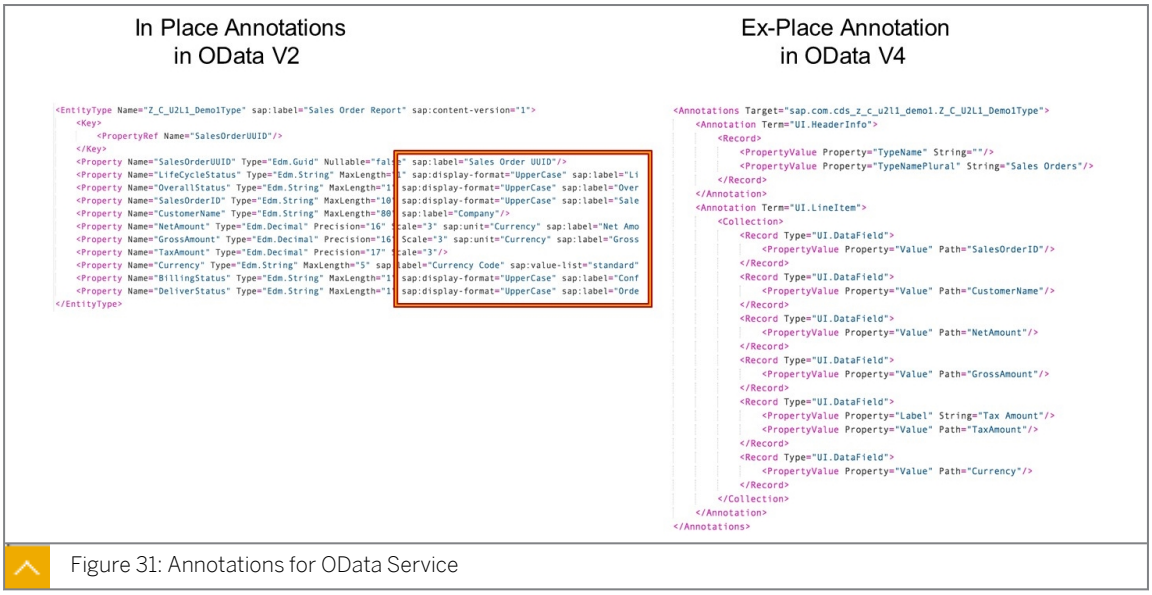


Figure 31: Annotations for OData Service

Annotations are descriptive information for OData Service.

In OData version 2.0, annotation is stored in XML tag of OData metadata, to describe an entity or fields, like datatype, nullable, and so on. SAP has expanded the standard OData annotation to satisfy more complex usage. All SAP specified annotation are started with 'sap:', for example, 'sap:creatable', 'sap:updatable'. Those annotations do not affect your SAPUI5 application. However, If you create your project from a wizard like '\CRUD Application', the code generated is affected by those annotations. For example, the editable property of fields

© Copyright. All rights reserved.

in the form generated for updating a record is determined by 'sap:updatable' annotation of corresponding field.

In OData version 4.0, annotations are separated from the main part of the OData metadata document. It can be a stand alone part of OData metadata document apart from main part, or stored in a separate file. This helps us to build more complex annotations, or write more than one annotation for one an OData service to satisfy different requirements.

Figure 32: Annotation Concepts (1): Vocabulary

Vocabulary defines how can you add annotations to an OData service. All annotation files must start from references and give an alias to some vocabularies.

The form of vocabulary is an XML document containing information of Terms and Types.

In an SAP Fiori elements scenario, the most important vocabulary is UI vocabulary, it defines most of annotations you can use in SAP Fiori elements. Other vocabularies, like Common and Communication, define annotations in specific domains and are used in several scenarios.

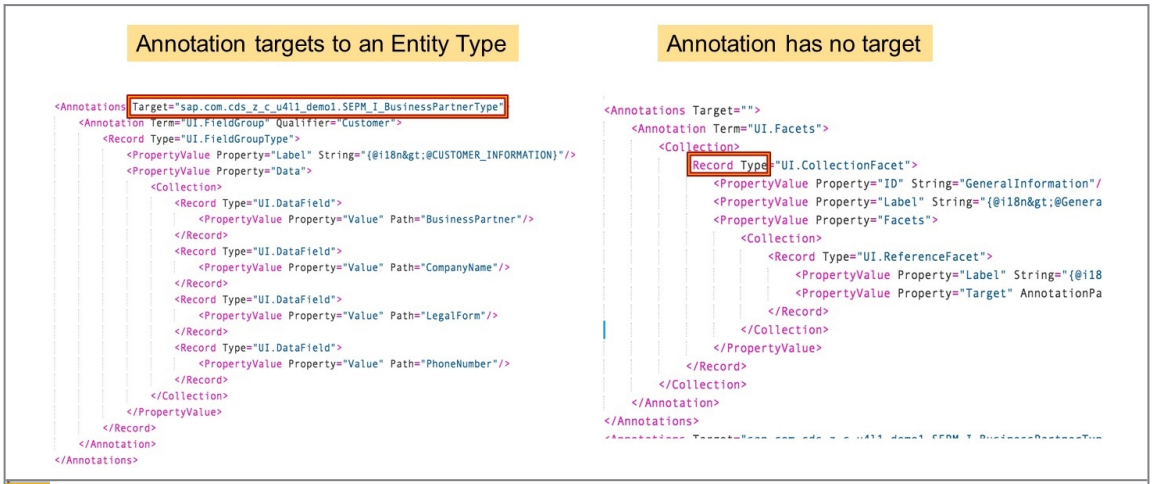


Figure 33: Annotation Concepts (2): Target

Annotations must grouped by targets. Targets associate a group of annotations to something in OData service. As most annotations are data relevant, they try to target an entity type or property. In some cases, annotations target an Association, a Function Import, or other objects in OData service.

Some annotations are nothing about data, or do not bind with particular entity type, those annotations do not have a target, they should be contained in a group of annotations with a blank "" target.

© Copyright. All rights reserved.

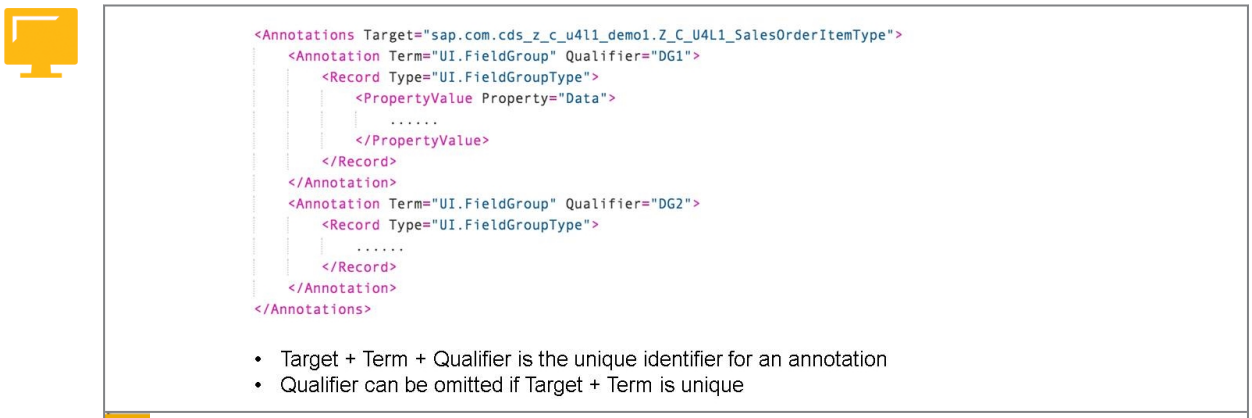


Figure 34: Annotation Concepts (3): Term and Qualifier

All annotations must have a term. A term determines the meaning of an annotation. For example, term UI.FieldGroup defines a group of fields.

If you have two annotations with the same term for the same target, a qualifier must be added to make your annotation unique.

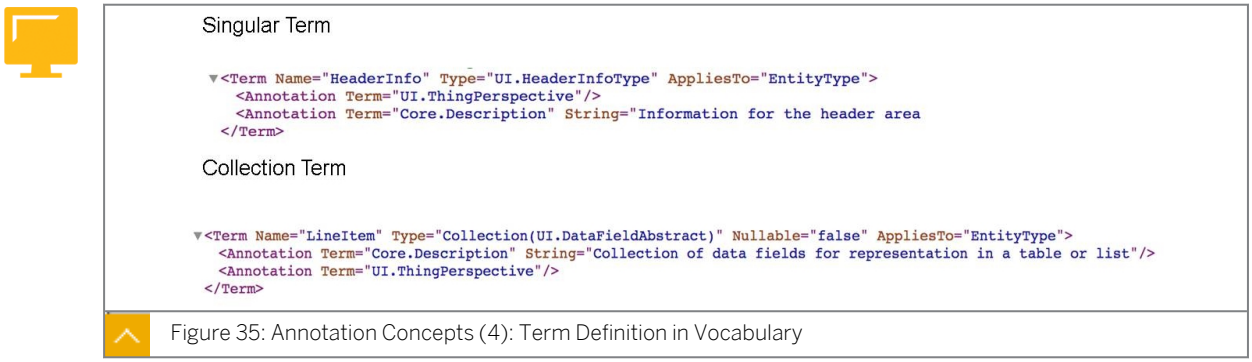


Figure 35: Annotation Concepts (4): Term Definition in Vocabulary

Terms are defined in a vocabulary. The following Information is part of a term definition:

- Term Name: The name of the term.
- Data Type: If there is only one record in the term, write the name of the type. If the term may have multiply records, the Type attribute needs to be 'Collection(<Data Type>)'
- Applies To: Defines which kinds of elements in OData service can be target of this term.
- Nullable: Default is true. Most annotations are nullable.
- Annotations to this term, used for development tools.

© Copyright. All rights reserved.

```
<ComplexType Name="HeaderInfoType">
  <Property Name="TypeName" Type="Edm.String" Nullable="false">
    <Annotation Term="Core.IsLanguageDependent"/>
    <Annotation Term="Core.Description" String="Name of the main entity type"/>
  </Property>
  <Property Name="TypeNamePlural" Type="Edm.String" Nullable="false">
    <Annotation Term="Core.IsLanguageDependent"/>
    <Annotation Term="Core.Description" String="Plural form of the name of the main entity type"/>
  </Property>
  <Property Name="Title" Type="UI.DataField" Nullable="false">
    <Annotation Term="Core.Description" String="Title, e.g. for overview pages"/>
  </Property>
  <Property Name="Description" Type="UI.DataField" Nullable="true">
    <Annotation Term="Core.Description" String="Description, e.g. for overview pages"/>
  </Property>
  <Property Name="ImageUrl" Type="Edm.String" Nullable="true">
    <Annotation Term="Core.IsURL"/>
    <Annotation Term="Core.Description" String="Image URL for an instance of the entity type. If the prope
    available or not valid the property TypeImageUrl can be used instead."/>
  </Property>
  <Property Name="TypeImageUrl" Type="Edm.String" Nullable="true">
    <Annotation Term="Core.IsURL"/>
    <Annotation Term="Core.Description" String="Image URL for the entity type"/>
  </Property>
</ComplexType>
```

Figure 36: Annotation Concepts (5): Complex Type

Most terms are described by a complex type. A complex type can have properties, which are further described by other complex types, or OData built in types (Edm.*) like string,int32,boolean.

```
<ComplexType Name="DataFieldAbstract" Abstract="true">
  <Property Name="Label" Type="Edm.String" Nullable="true">
    <Annotation Term="Core.Description" String="A short, human-readable text suitable for labels and captions in UIs"/>
    <Annotation Term="Core.IsLanguageDependent"/>
  </Property>
  <Property Name="Criticality" Type="UI.CriticalityType" Nullable="true">
    <Annotation Term="Core.Description" String="Criticality of the data field value"/>
  </Property>
  <Property Name="CriticalityRepresentation" Type="UI.CriticalityRepresentationType" Nullable="true">
    <Annotation Term="Core.Description" String="Decides if criticality is visualized in addition by means of an icon"/>
  </Property>
  <Property Name="IconUrl" Type="Edm.String" Nullable="true">
    <Annotation Term="Core.Description" String="Optional icon to decorate the value"/>
    <Annotation Term="Core.IsURL"/>
  </Property>
</ComplexType>
```

Type “DataFieldWithAction”has all properties of “ DataFieldAbstract”and additional property ”Action”

```
<ComplexType Name="DataFieldWithAction" BaseType="UI.DataField">
  <Annotation Term="Core.Description" String="The action is tied to
  to a specific data value."/>
  <Property Name="Action" Type="Common.QualifiedName" Nullable="fals
    <Annotation Term="Core.Description" String="Qualified name of an
  </Property>
</ComplexType>
```

Figure 37: Annotation Concept (6): Type Inheritance

For better re-usability, inheritance of types is supported.

In this example, DataFieldAbstract is an abstract data type (Abstract = true) and type DataFieldWithAction has a base type of DataFieldAbstract, so the latter have properties of the base type, with additional properties defined for itself.

© Copyright. All rights reserved.

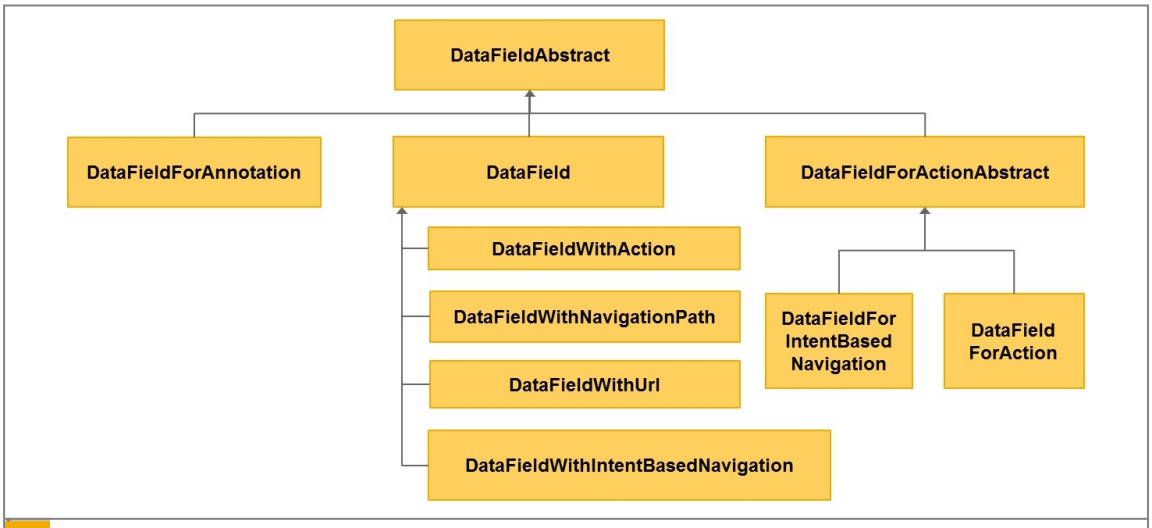


Figure 38: Important Data Type: DataField* Series

A collection of data types starting with DataField are the foundation of UI annotations. Most terms and complex types have relationships with those datatypes since data fields are what end user sees, selects, and searches. All types are inherit from DataFieldAbstract.

The following list provides the definitions for those types:

DataFieldAbstract

Abstract type for all DataField* types, defines common properties to describe a field.

DataField

A normal data field on screen, displays values from a field of an entity type of the OData service.

DataFieldForAnnotation

Another annotation determines the field displayed on screen.

DataFieldWithAction

Rendered as a button or link, the actual action is relevant to the value of the field.

DataFieldForAction

Rendered as a button or link, the action is not relevant to value of the field in contrast to DataFieldWithAction.

DataFieldWithNavigationPath

The data source of the field is a field in other entity types which can be accessed through an association.

DataFieldWithURL

The data field is rendered as hyperlink and opens a URL when selected by the user.

DataFieldWithIntentBasedNavigation

The data field is rendered as a hyperlink, jumps to an intent defined in SAP Fiori Launchpad, the target is relevant to value of the field.

DataFieldForIntentBasedNavigation

The data field is rendered as hyperlink, jumps to an intent defined in SAP Fiori Launchpad, the target is not relevant to the value of the field in contrast to DataFieldWithIntentBasedNavigation.

© Copyright. All rights reserved.

Smart Control, Concept

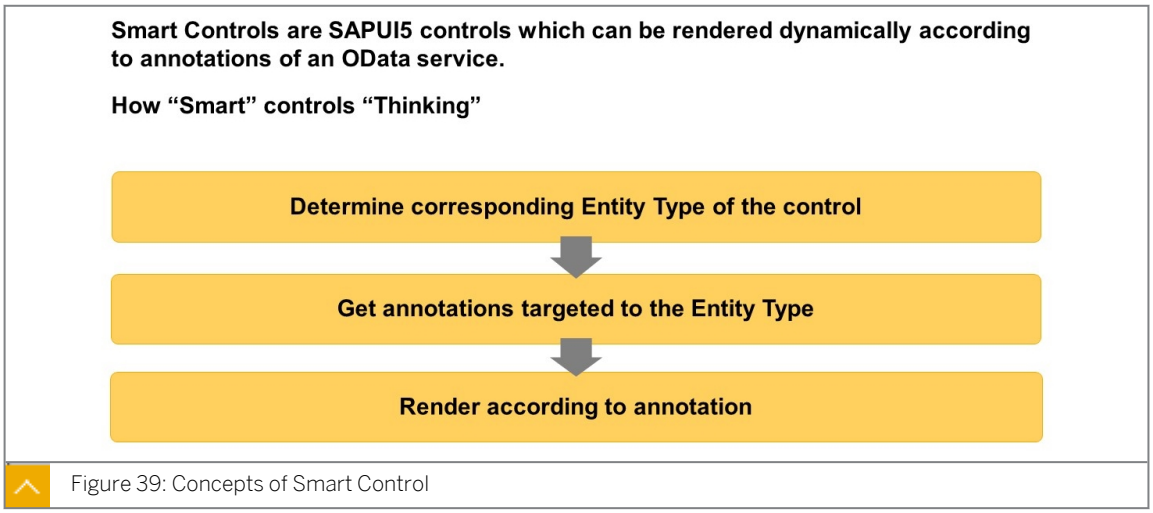


Figure 39: Concepts of Smart Control

Smart Controls are a collection of controls developed before SAP Fiori elements. The initial purpose is to replace basic controls in SAPUI5 in most common use cases.

The idea was to build a whole application by smart controls, this was a Smart Template, the predecessor of SAP Fiori elements. Developers can use those control standalone, or use Smart Template to build smart applications. The development and maintenance of Smart Controls and SAP Fiori elements are separate.

Currently, Smart Control is mainly for SAP Fiori elements, and not recommend for standalone usage. All new features of smart controls will be determined by requirements of SAP Fiori elements.

Smart Control Example

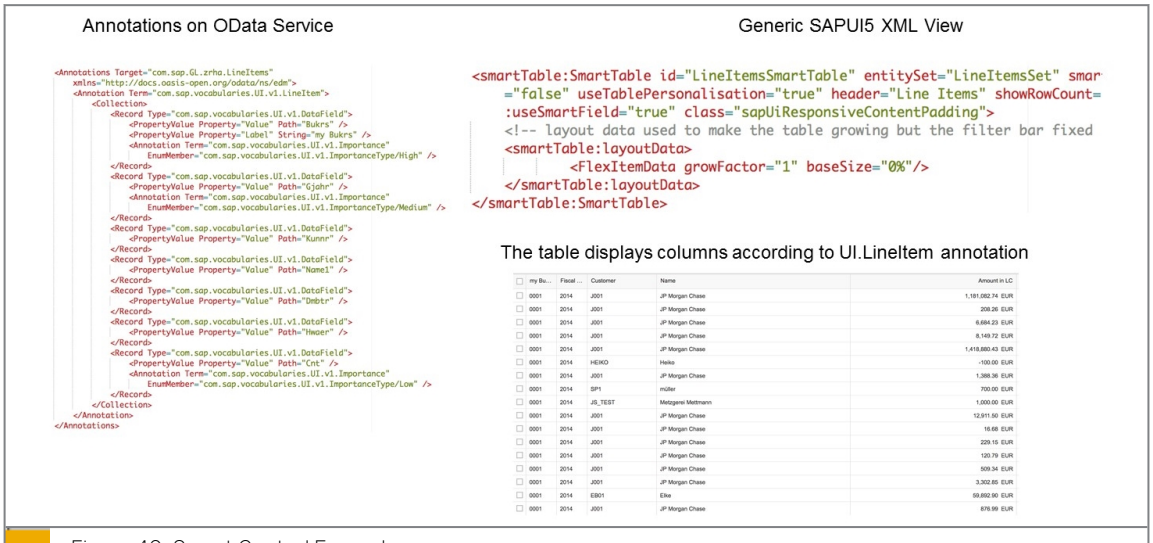


Figure 40: Smart Control Example

The figure, Smart Control Example, shows an example of a smart control.

© Copyright. All rights reserved.

LESSON SUMMARY

You should now be able to:

- Explain the concept of Fiori Elements
- Explain the concept of annotations of OData services
- Explain the concept of Smart Control

© Copyright. All rights reserved.

Explaining Templates for Fiori Elements

LESSON OBJECTIVES

After completing this lesson, you will be able to:

- Explain Templates for Fiori Elements

Templates of SAP Fiori Elements

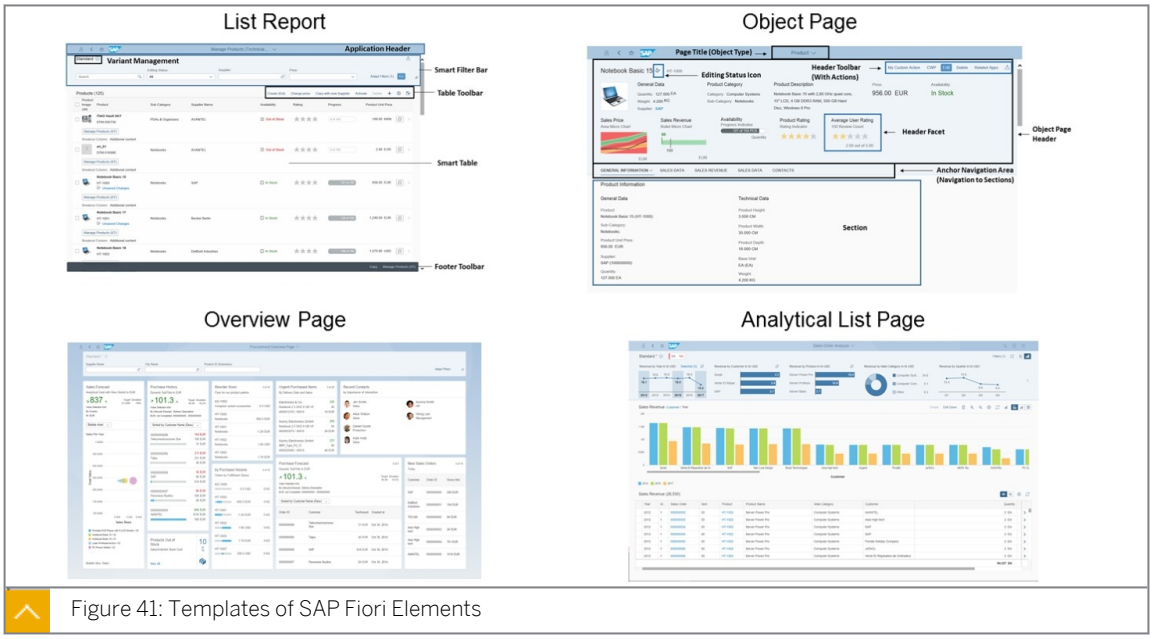


Figure 41: Templates of SAP Fiori Elements

The following list outlines the current templates for SAP Fiori elements:

- List Report
- Object Page
- Analytic List Page
- Overview Page

© Copyright. All rights reserved.

List Report

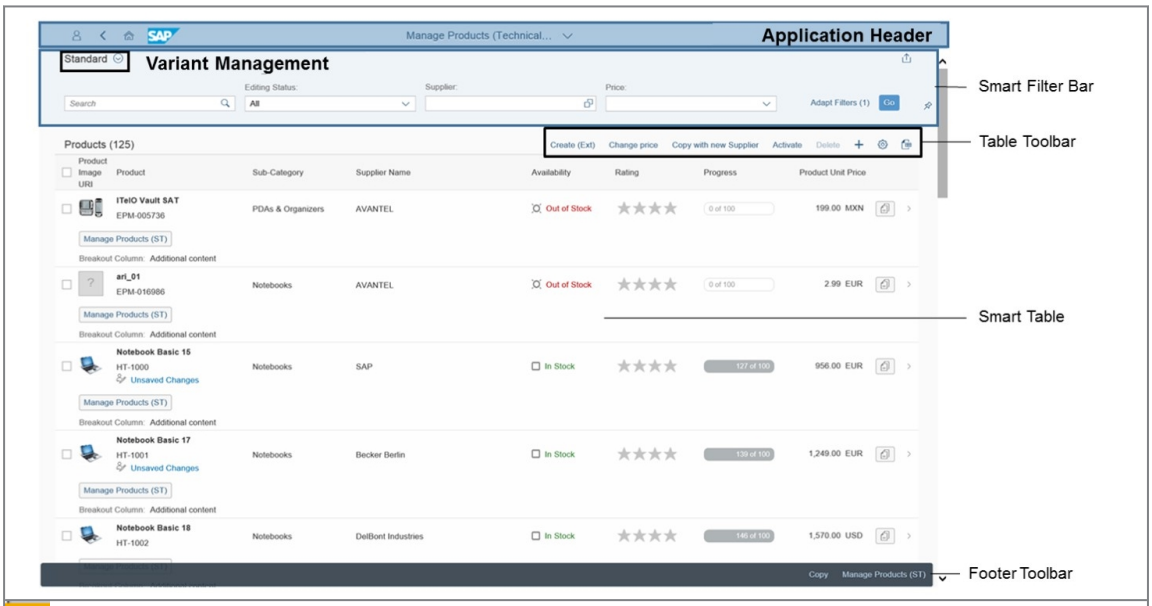


Figure 42: List Report

The list report allows the user to work with a large list of items. It combines powerful functions for filtering large lists with different ways of displaying the resulting item list.

List Report - Elements

The list report view includes the following main elements:

Application header

Smart filter bar with variant management and the generic Share menu that includes the following actions: Send Email.

Save as Tile

Share in SAP Jam (if integration with SAP Jam is configured).

By default, the smart filter bar is displays when users launch an app. When choosing *Go*, the content of the list report is displays. If users have set *Execute on Select* for their default variant, the content of the list report is displays immediately upon launching the app, and the smart filter bar is collapsed.

Smart table

Footer toolbar

Can include optional actions.

A List Report can display data not only as plain text, but also various other forms to improve user experience. It also supports editable cell and can leverage create, update, and delete operation to corresponding back end code.

From that, we can use the List Report as replacement of ALV in SAP GUI. Lots of ABAP queries created by SQVI and ALV based ABAP reports can be replaced by List Report.

© Copyright. All rights reserved.

Object Page

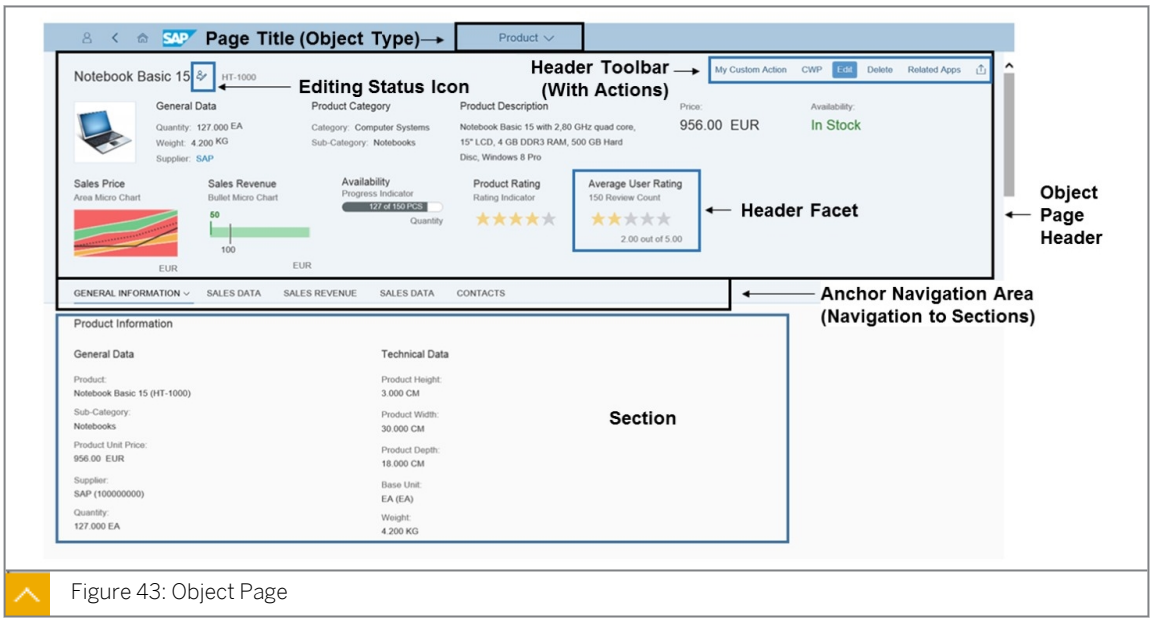


Figure 43: Object Page

The Object page displays information of a single business entity. A major entry of object page, is to select a row in list report, so this page is combined in List Report template. When you create an SAP Fiori elements application using List Report, you get an Object page.

Object Page - Elements

The object page is made up of the following elements:

- Page title, which is set to the object type or product, for example, product
- Object header including the following:
  - Title and subtitle
  - Editing status icon
- Header toolbar, containing generic actions (in Display mode)
- Optional elements, which can include the following:
  - A description
  - An image of the object
  - Buttons in the header toolbar for use case-specific actions, for example, Edit and Delete
- Header facets to showcase important information for the object. Header facets can contain, for example:
  - Label-field pairs, to show, for example, price or availability. We recommend using a maximum of five label-field pairs.
  - Smart controls, to show, for example, a micro chart detailing sales revenue, or a rating indicator to visualize the average of all user-assigned ratings.
- Anchor navigation area that lets users navigate to the individual content area sections

© Copyright. All rights reserved.

- Content area, in which data is organized into sections that can contain field groups or a table
- Footer bar (not shown), in which actions and the Show Messages button are available (if applicable). The footer bar of sub-item object pages also includes the *Apply* button, in create and edit mode. This action concludes the current create or edit activity, saves the draft, and navigates one step up in the object hierarchy. A toast message displays if the operation was successful.

Overview Page

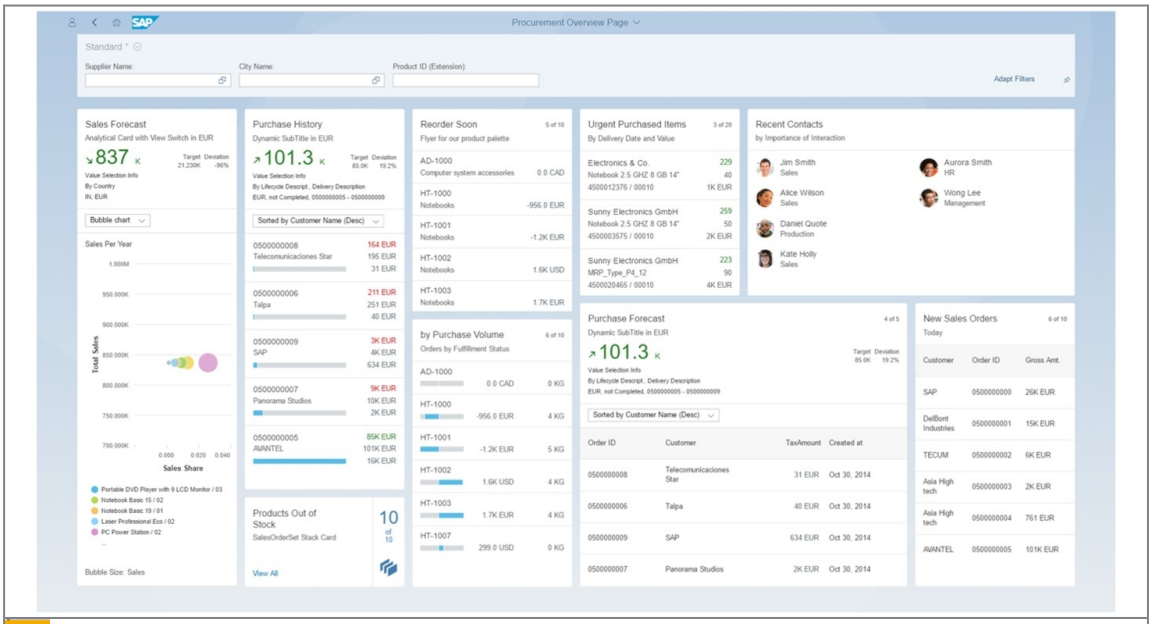


Figure 44: Overview Page

An Overview page is a data-driven SAP Fiori application built using SAPUI5 technology, OData services, and annotations for organizing large amounts of information.

The Overview page provides a quick access to vital business information at a glance, in the form of visual, actionable cards. The user-friendly experience makes viewing, filtering, and acting upon data quick and simple. While simultaneously presenting the big picture at a glance, business users can focus on the most important tasks enabling faster decision making as well as immediate action.

The application lets you create several cards for different types of content that helps in visualizing information in an attractive and efficient way. You can create overview pages and add cards to the page using the overview page wizard in SAP Business Application Studio.

The displayed data is fully interactive, with clickable areas for easy navigation to relevant applications. Based on SAP Fiori, overview pages organize action items with a fully responsive user interface. Users can access overview pages from SAP Fiori launchpad and narrow down the information displayed, or opt to hide cards to focus on a particular topic.

Overview Page - Components

The overview page application contains the following main components:

Application header:

© Copyright. All rights reserved.

Provides a description of the area for which this application provides an overview (for example, procurement or sales). From the header area, users can change user account settings and manage cards (show/hide).

Smart filter:

Provides application-level filters for changing the levels of data displayed in the cards. For example, you could use the filter to display only transactions larger than $10,000, only items lighter than 50kg, and so on.

Cards:

A card is a smart component that uses UI annotation to render its content. Each card is bound to a single entity set in a data source. A card may display a donut or bar chart, or a table. Stack cards contain a set of quick view cards which can be viewed in an object stream. Cards are displayed on the overview page in up to five responsive columns and can be rearranged by dragging and dropping.

Analytical List Page

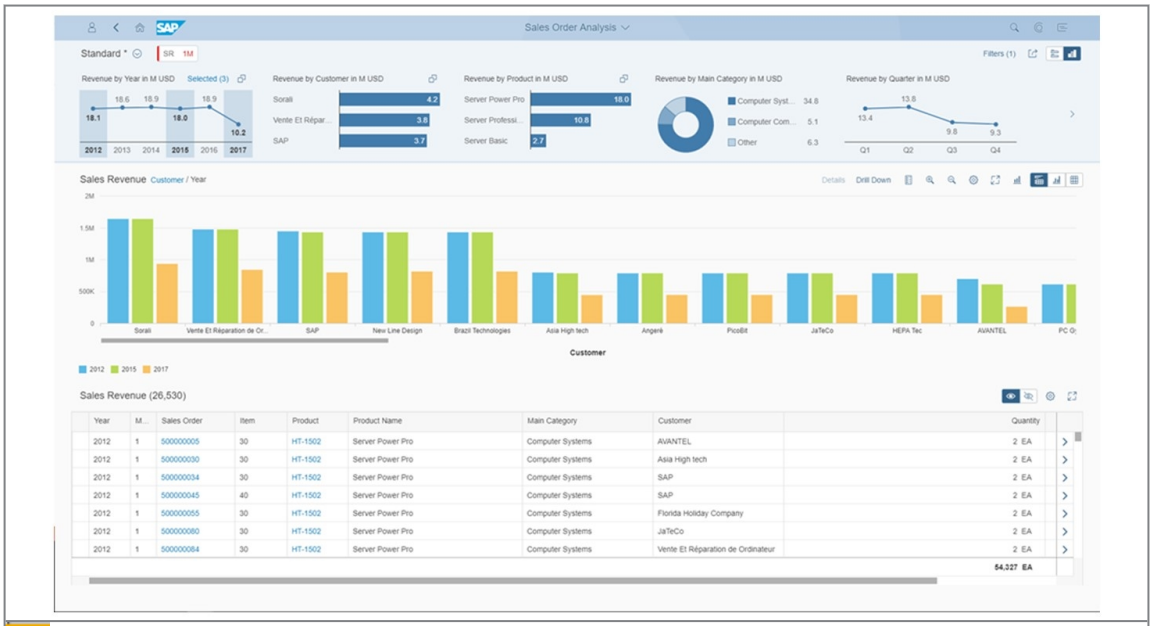


Figure 45: Analytical List Page

The Analytical List Page (ALP) is an SAP Fiori elements application for detailed analytics.

This is a brand new template introduced in SAP S/4HANA 1709 and ABAP 7.52.

It lets you analyze data from different perspectives, to investigate a root cause, and to act on transactional content. You can identify relevant areas within data sets or significant single instances using data visualization and business intelligence. All this can be done seamlessly within one page.

The combination of transactional and analytical data using chart and table visualization lets you view relevant data quickly. This hybrid view of data allows an interesting interplay between the chart and table representations.

Configure ALP to include the following use cases seamlessly within one page:

- Related KPIs (key performance indicators) on the header area as KPI tags.
- These KPI tags further allow a progressive disclosure and navigation through KPI cards.

© Copyright. All rights reserved.

- Filter data sets used for the main content area through different filter modes.

  For example, visual filters provide an intuitive way of choosing filter values from an associated measure value.

- Seamless navigation to applications from the content area and KPI card area.
- Customizing and sharing ALP as a page variant with other users.

LESSON SUMMARY
You should now be able to:

- Explain Templates for Fiori Elements

© Copyright. All rights reserved.

Exploring the Development Environment

LESSON OBJECTIVES

After completing this lesson, you will be able to:

- Explore the Development Environment

The Development Environment

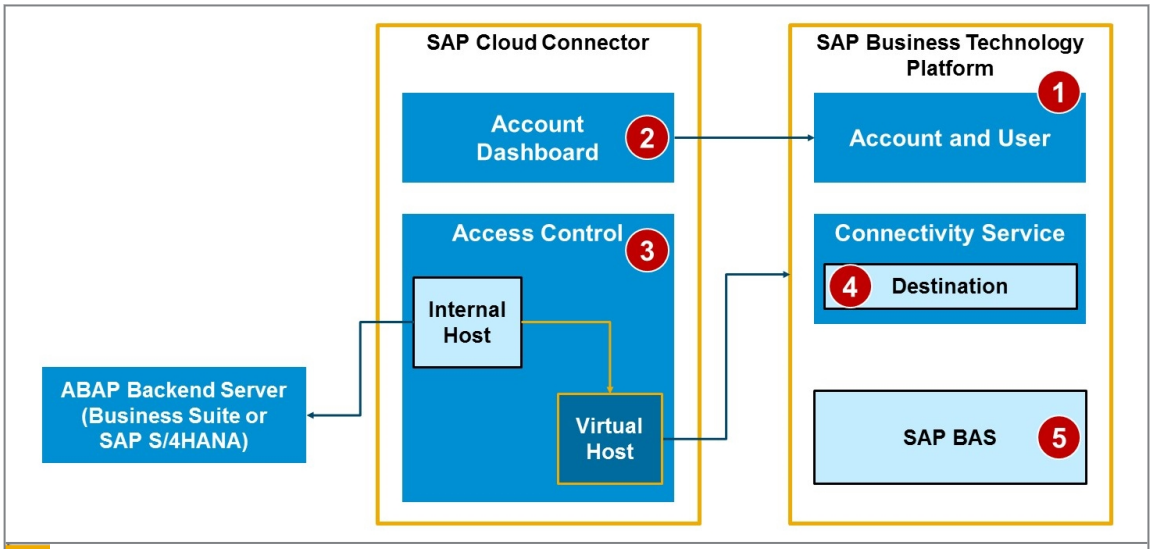


Figure 46: Set Up Your Development Environment

To set up development for SAP Fiori elements developments, you need:

- An SAP BTP account to run development tools – SAP Business Application Studio.
- Server plays the role of OData backend, in this course, we are using the embedded scenario. So we use only one ABAP server for backend logic, and SAP Gateway server for publishing OData service and SAP Fiori application.
- SAP Cloud Connector to set up a secure connection between your Gateway server and SAP BTP.
- If you develop in the BTP with the help of the BAS we also need a DEV Space.

Please see the SAP Fiori Deployment Options and System Landscape Recommendations what infrastructure setup fits your needs.

Major steps of configuration are:

1. Register an account in SAP Business Technology Platform.
2. Add your SAP BTP account into SAP Cloud Connector.

© Copyright. All rights reserved.

3. Configure access control information in SAP Cloud Connector to connect your internal SAP Gateway server and expose it through a virtual host name/port.
4. Create a destination in connectivity service of SAP BTP. Use this destination to connect to your internal SAP Gateway through virtual host name defined in SAP Cloud Connector.
5. Test your configuration by running wizard of SAP BAS. Select the destination you created and see if a list of OData services are shown up.

LESSON SUMMARY

You should now be able to:

- Explore the Development Environment

© Copyright. All rights reserved.

Unit 2
Lesson 4

Exploring the Basic Process of Building Fiori Elements Application

LESSON OBJECTIVES

After completing this lesson, you will be able to:

- Explore the basic process of building Fiori Elements applications

Building SAP Fiori Elements Applications Process

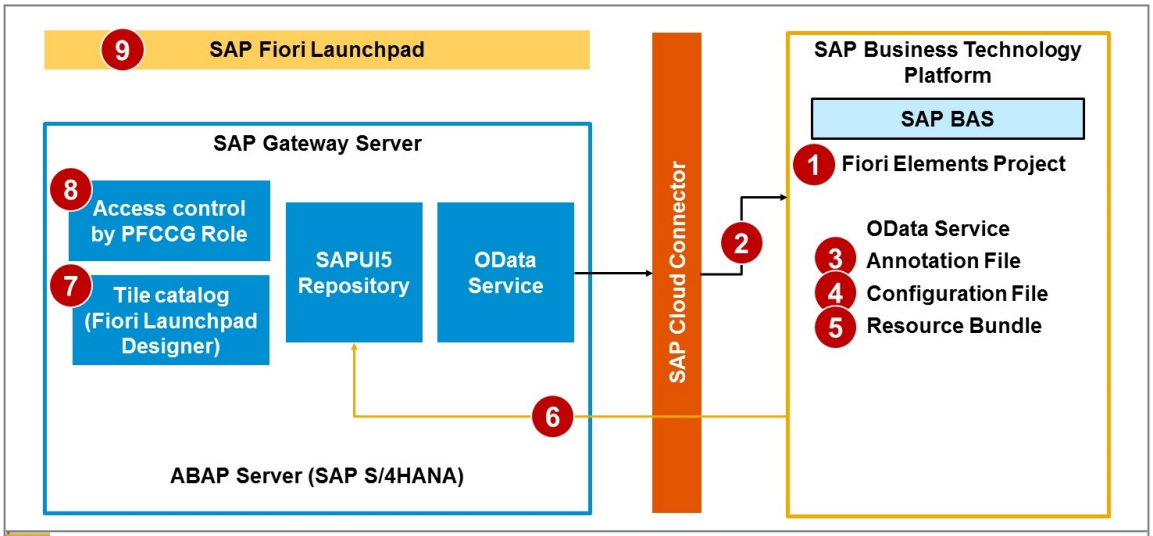


Figure 47: Overall Process of Development

1. Create an SAPUI5 project from an SAP Fiori elements template.
2. Choose an OData service as data provider of your project.
3. Create an annotation file associated to the OData service and write annotation in it.
4. Modify configuration file to customize your application(Optional).
5. Translate resource bundles (i18n* files) to make your application support multiple languages (Optional).
6. Deploy your SAP Fiori elements application to SAPUI5 repository.
7. Create semantic object, tile catalog, target mapping, tiles, and other configuration for SAP Fiori Launchpad.
8. Add catalog to user menu in a PFCG role, then grant the role to user.
9. Log on to SAP Fiori Launchpad, add the tile to Launchpad.

© Copyright. All rights reserved.

Note:
The process is not 100% the same for all templates. Here, we only take List Report as an example. Process for other templates will be discussed in later units.

The above figure shows the setup for embedded deployment. If hub deployment is in use, the yellow boxes are part of the front end server (FES).

1. Create a SAPUI5 Project

Figure 48: 1. Create an SAPUI5 project

To create an SAP Fiori elements project, you need to create an SAP Fiori Elements project. Choose the appropriate templates for your use case.

Choose the Data Source and den OData service. Select the Entity and, if available, the navigation entity. Provide Project Attributes like the name of the module, application title, and application namespace.

2. Choose OData Service

To bind your SAP Fiori elements to an OData Service, do the following:

1. Choose Destination and OData Service.
2. If an annotation file for this OData service already exists, you can see it in the Review Existing Annotation Files step. You can also add the annotation file here, or in a later step.

   SAP provides you with the most flexibility for creating an annotation file, you can create it either in the back end (External Annotation) or front end (Local Annotation).
3. In the step, Basic Customizing, you need to provide basic information for the template. The content depends on the template you choose.

3. Create Annotation File and Writing Annotation

In SAP Business Application Studio, a project of type SAP Fiori elements contains an annotation file. To add annotations, the SAP BAS provides the Guided Development feature.

© Copyright. All rights reserved.

Guided Development

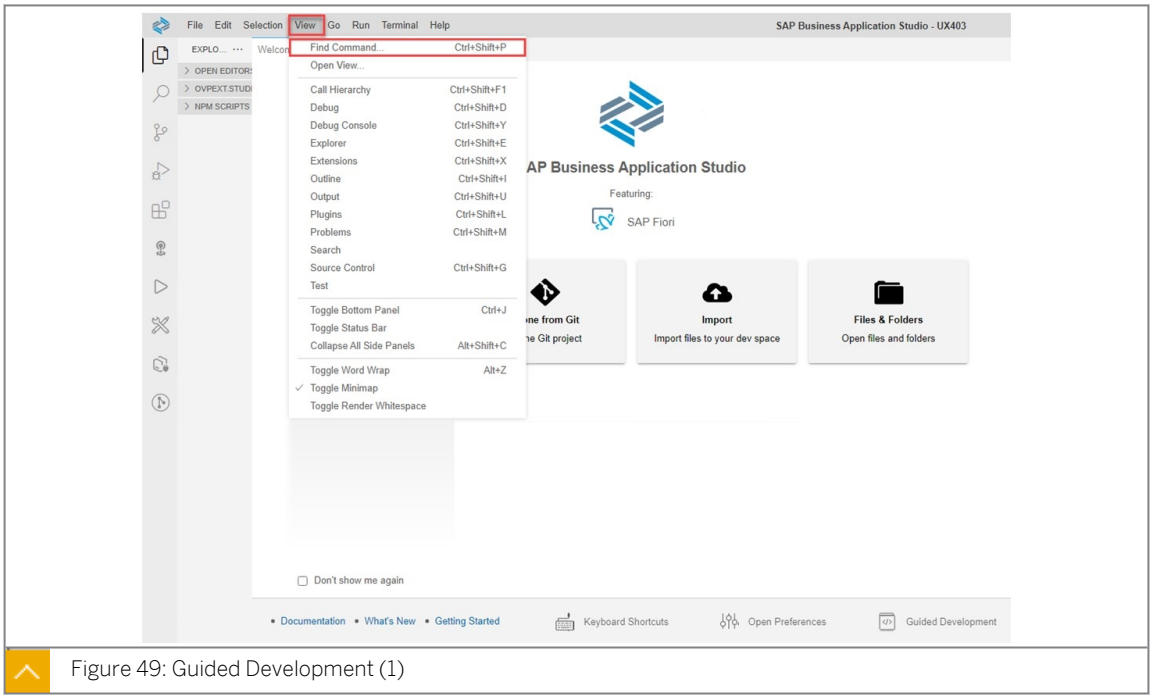


Figure 49: Guided Development (1)

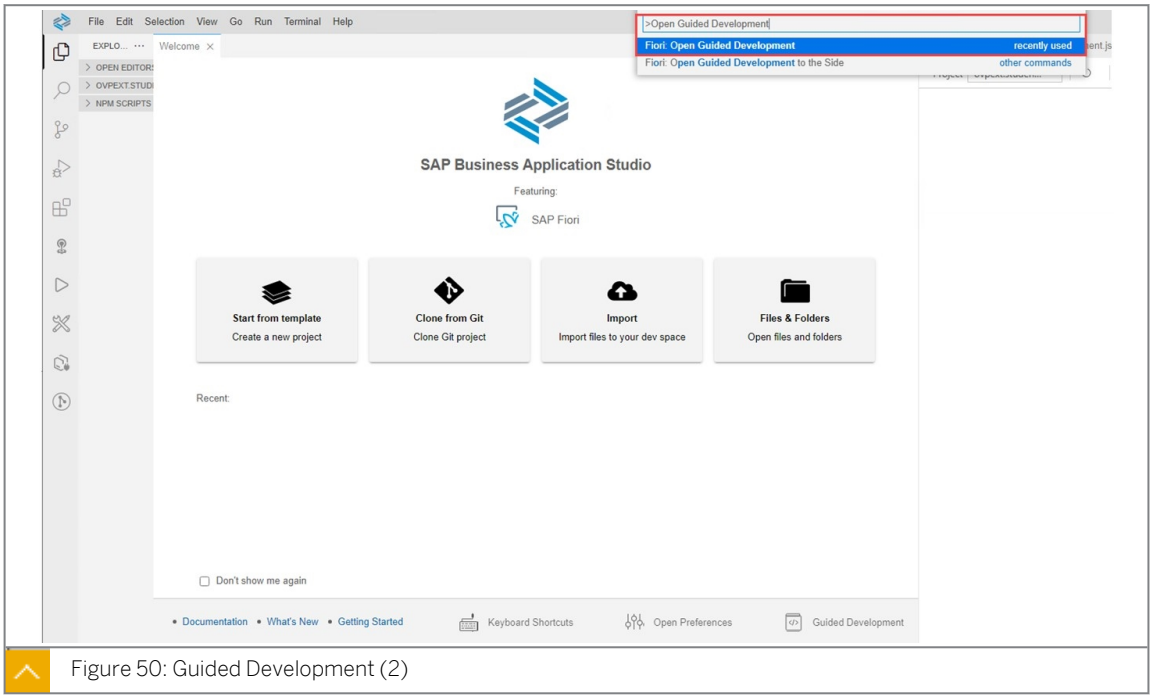


Figure 50: Guided Development (2)

© Copyright. All rights reserved.

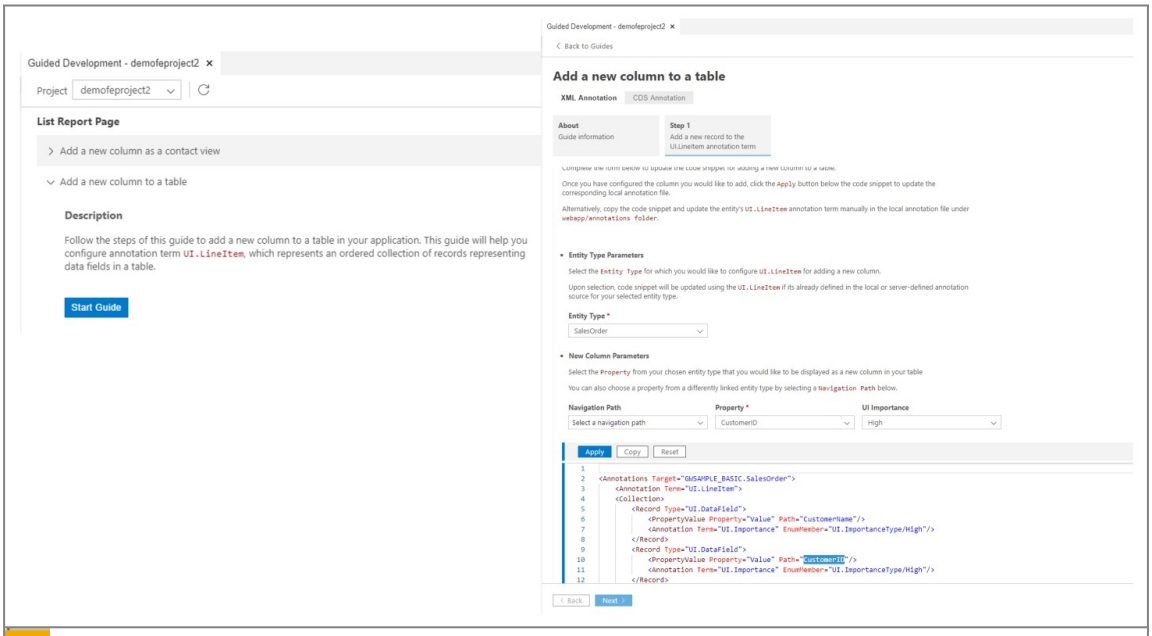


Figure 51: Guided Development (3)

The Guided Development feature of SAP BAS guides the developer adding new annotations to the annotation file.

4. Modify Configuration File

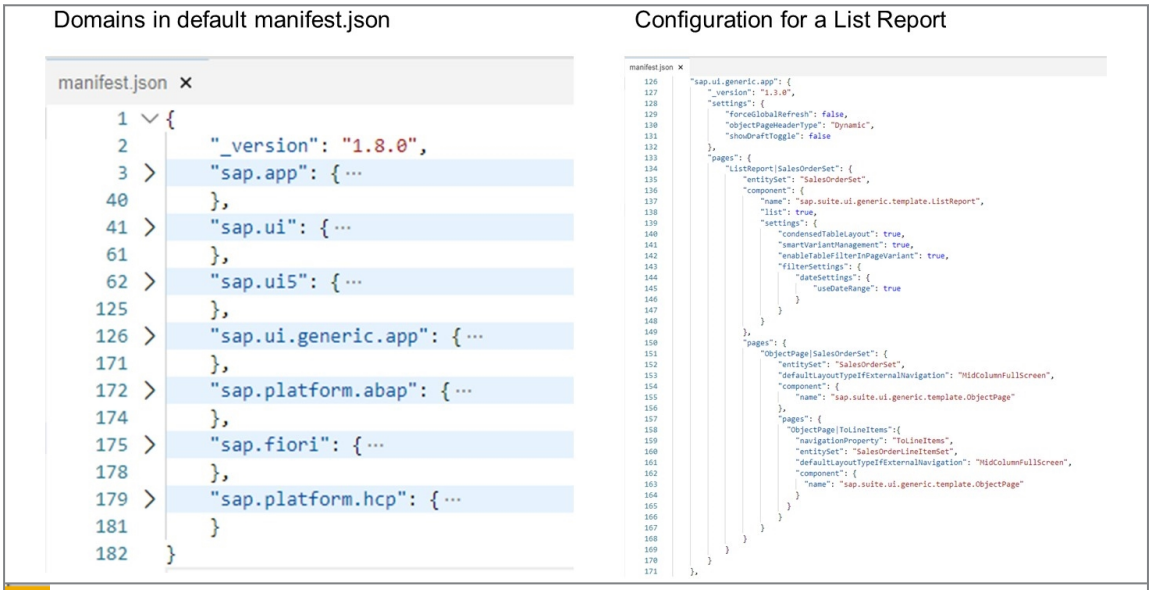


Figure 52: 4. Modify Configuration File to Customize Your Application (Optional)

All information you provided in the wizard will result as configuration information in manifest.json. The file is located in the webapp folder in your project. You can navigate detailed information by double-clicking it and choosing *Code Editor*.

The configurations are divided into several domains. Domain “sap.app” has the configuration information generally used by all SAPUI5 application, like name, title, OData service, and location of its annotations. Based on the template you choose, it may have other specific domains contain other information. Fox example, domain “sap.ui.generic.app” contains configuration information about the List Report and Object Page.

© Copyright. All rights reserved.

When you need to change some basic information, like OData service name or entity set name, you can modify this file directly. In some cases, advanced features also need some adjustments in this file.

5. Translate Resource Bundles

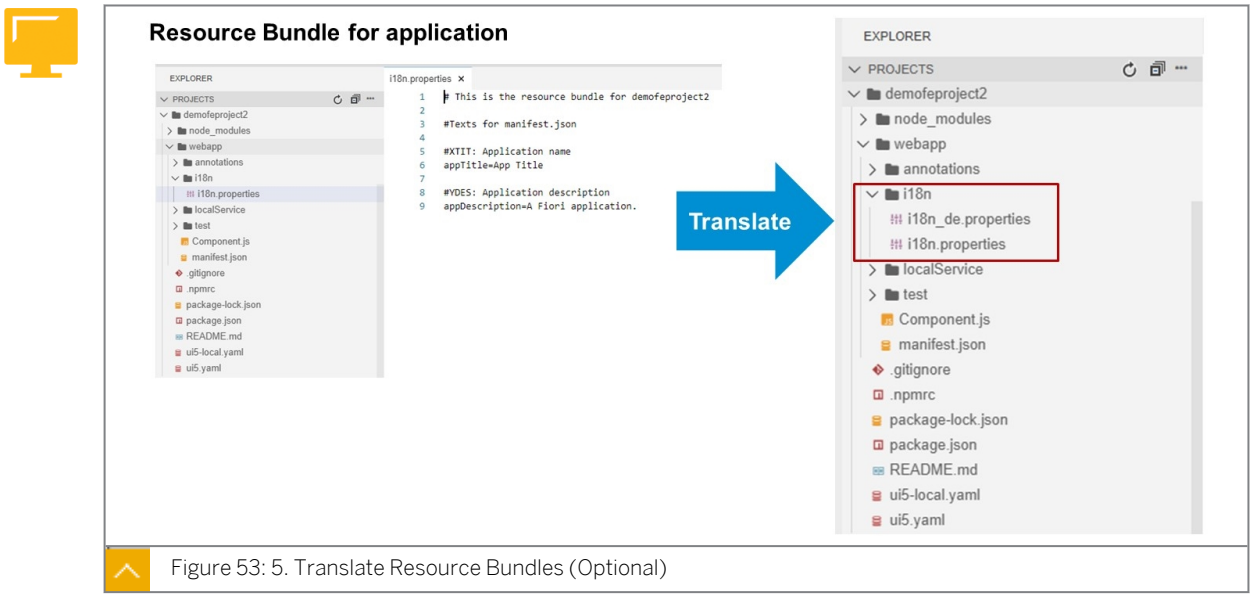


Figure 53: 5. Translate Resource Bundles (Optional)

For the need of internationalization of static text in your application. It's best to store static text content as a key-value pair in a file. In SAP Fiori elements, the template will create a default resource bundle file (i18n.properties) for the application and each page in it. You can translate each file into any language by creating a i18n_<language iso code>.properties in the same position of the resource bundle you want to translate.

6. Deploy

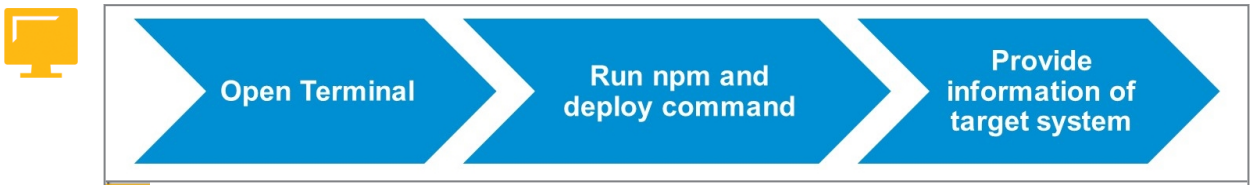


Figure 54: 6. Deploy

To deploy your SAP Fiori elements to a server. You should complete following steps:

1. In BAS, open a terminal window. You can use the button in the BAS Terminal to select a new Terminal.
2. Enter the command npm run deploy-config in the console.

© Copyright. All rights reserved.

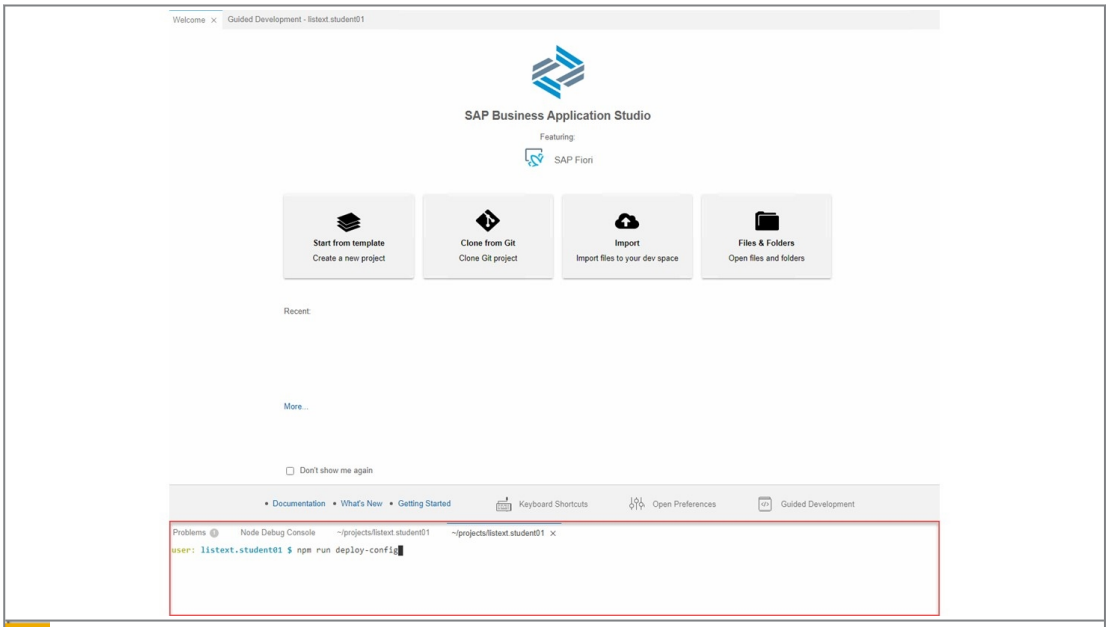


Figure 55: Terminal Command

3. Provide information, like program name, description, and package.
4. To deploy the application enter the command npm run deploy

```
{
  "deploy": {
    "sourceFolder": "./dist",
    "targetSystem": "http://abc.com",
    "destinationName": "<destination name>",
    "client": "010",
    "username": "<username>",
    "password": "<password>",
    "appName": "foo",
    "description": "my description",
    "abapPackage": "myPackage",
    "abapTransport": "myTransport"
  },
  "cli": {
    "interactiveFallBack": true
  }
}
```

Figure 56: Deploy Your SAP Fiori Elements

© Copyright. All rights reserved.

7. SAP Fiori Launchpad Configuration

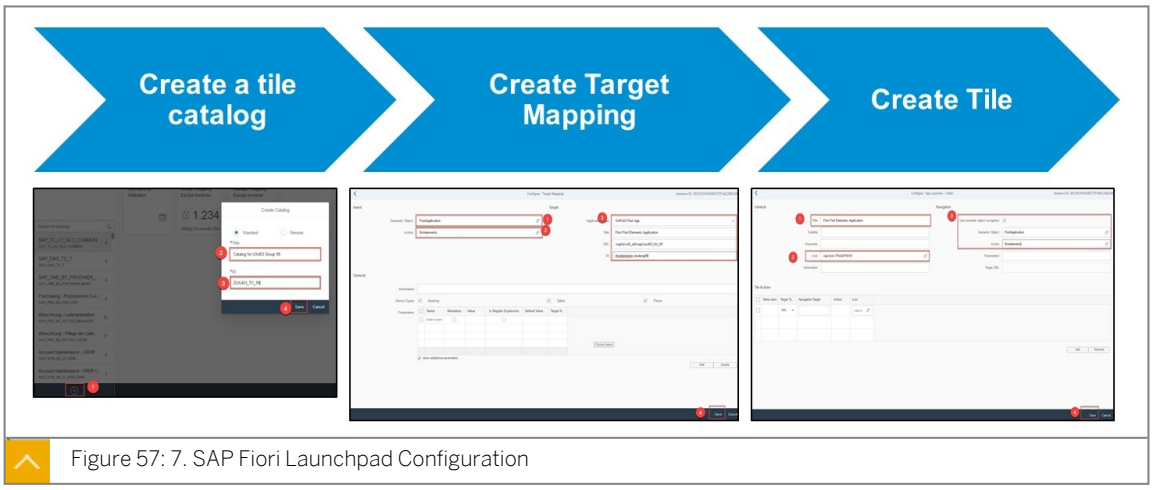


Figure 57: 7. SAP Fiori Launchpad Configuration

If the SAP Fiori elements application is deployed on Netweaver ABAP, you need to enable the application by the following steps:

1. Open SAP Fiori Launchpad Designer and create a tile catalog for the application.
2. Create a target mapping. Target mapping should have detailed technical information for the access of the application, and expose the application as an intent (Semantic Object - Action).
3. Create a tile based on the target mapping.

> Note:
> Detailed information of SAP Fiori Launchpad configuration is described in UX100.

8. Authorization

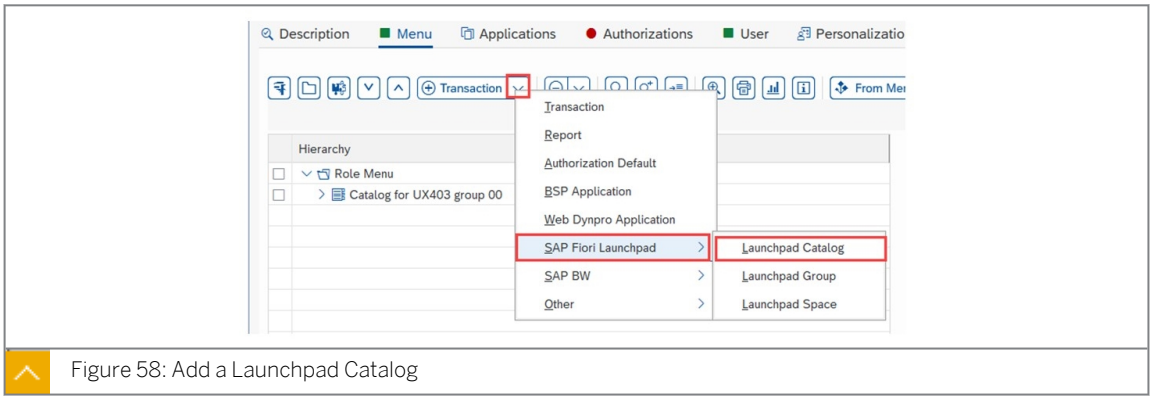


Figure 58: Add a Launchpad Catalog

To give a user permission to access the SAP Fiori application, several application-specific authorizations are needed.

For front end user:

The user must have a role which has the tile catalog in its role menu.

For back end user:

© Copyright. All rights reserved.

The user must have authorization object to access the OData service and business data.

9. Log on to SAP Fiori Launchpad, Add the Tile to Launchpad

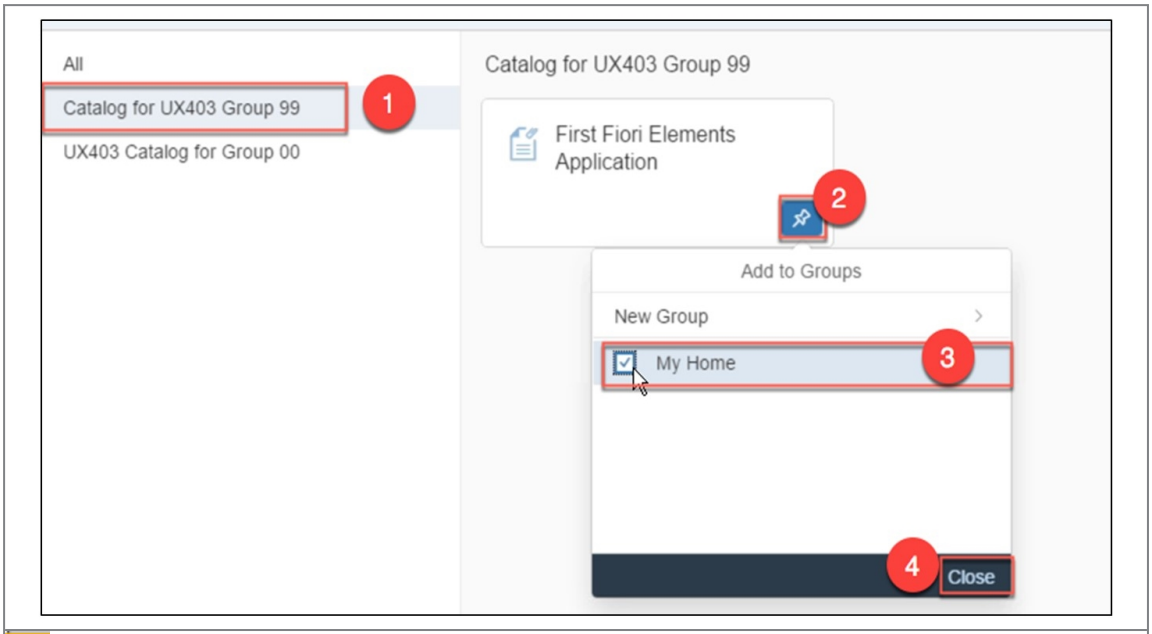


Figure 59: 9. Log on to SAP Fiori Launchpad, Add the Tile to Launchpad

Users can log on to SAP Fiori Launchpad and add tiles from catalog to their home page.

> Note:
> Detailed information of SAP Fiori Launchpad usage is described in UX100.

To Create an SAP Fiori Elements Application

1. Create an SAP Fiori elements application.
2. Choose an OData service.
3. Create an annotation file and writing annotation.
4. Modify the configuration file.
5. Translate resource bundles.
6. Deploy your SAP Fiori elements application.
7. Configure SAP Fiori Launchpad.
8. Authorize user to access the SAP Fiori application.
9. Log on to SAP Fiori Launchpad and add the tile to the SAP Fiori Launchpad.

LESSON SUMMARY

You should now be able to:

- Explore the basic process of building Fiori Elements applications

© Copyright. All rights reserved.

Unit 2
Lesson 5

Using the Core Data Services (CDS) View

LESSON OBJECTIVES
After completing this lesson, you will be able to:

- Use the CDS View and SADL

The Core Data Services (CDS) View

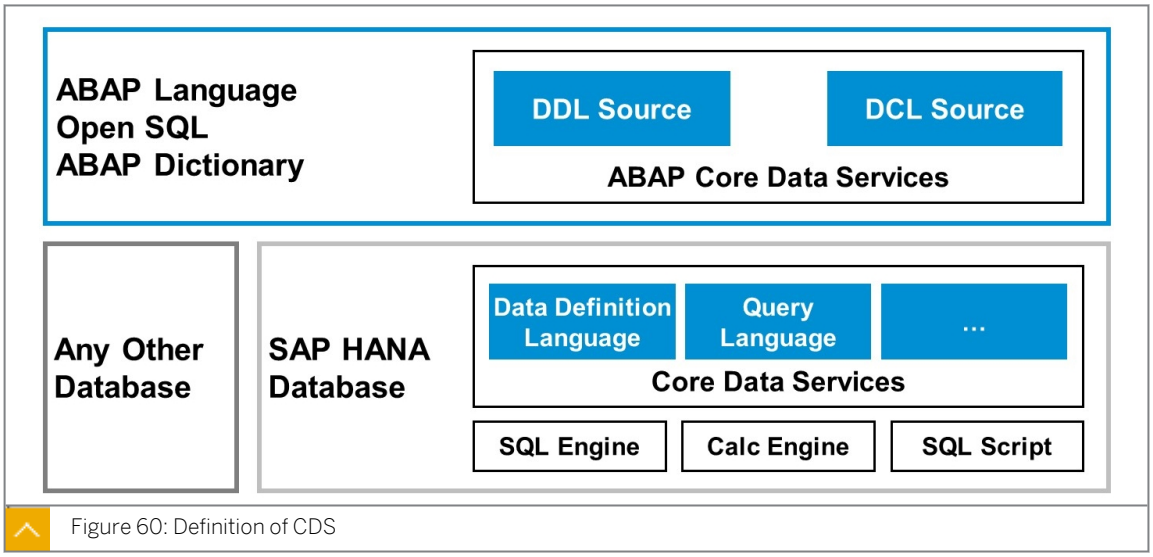


Figure 60: Definition of CDS

Note:
The purpose of this lesson is to let you have basic understand of CDS.

For detailed knowledge of CDS, please refer to course S4D430.

The ABAP Core Data Services (ABAP CDS) are the platform-independent implementation of the general CDS concept for AS ABAP. The ABAP CDS makes it possible to define semantic data models on the standard AS ABAP database. Unlike the SAP HANA-specific variant HANA CDS, the ABAP CDS are independent of the database system. The entities of the models defined in ABAP CDS provide enhanced access functions compared with existing database tables and views defined in ABAP Dictionary, making it possible to optimize Open SQL-based applications. This is particularly clear when an AS ABAP uses an SAP HANA database, since its in-memory characteristics can be implemented in an optimum manner.

© Copyright. All rights reserved.

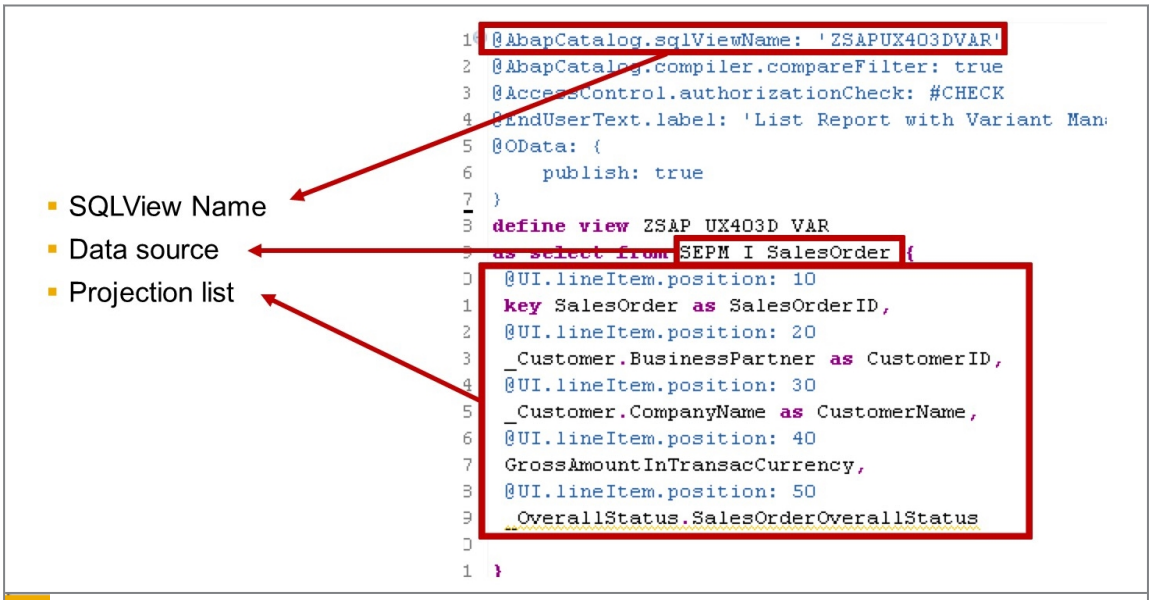


Figure 61: Basic Syntax of CDS

A basic CDS defines a view, the definition of CDS uses SQL like syntax.

SQL View Name: For compatibility with ABAP dictionary, each CDS must have a name in Old ABAP dictionary format (Max length:16).

Data source: A CDS view selects data from one or several database tables or CDS views. If data comes from more than one data source, SQL JOIN syntax can be used to select data from multiply data sources.

Projection List: The most important part of a CDS view is the projection list. In projection list, complex expression can be used to create calculated field.

Note:
The process of creating a CDS view is described in the exercise: Create a List Report using CDS with Annotation

Annotations for CDS Views

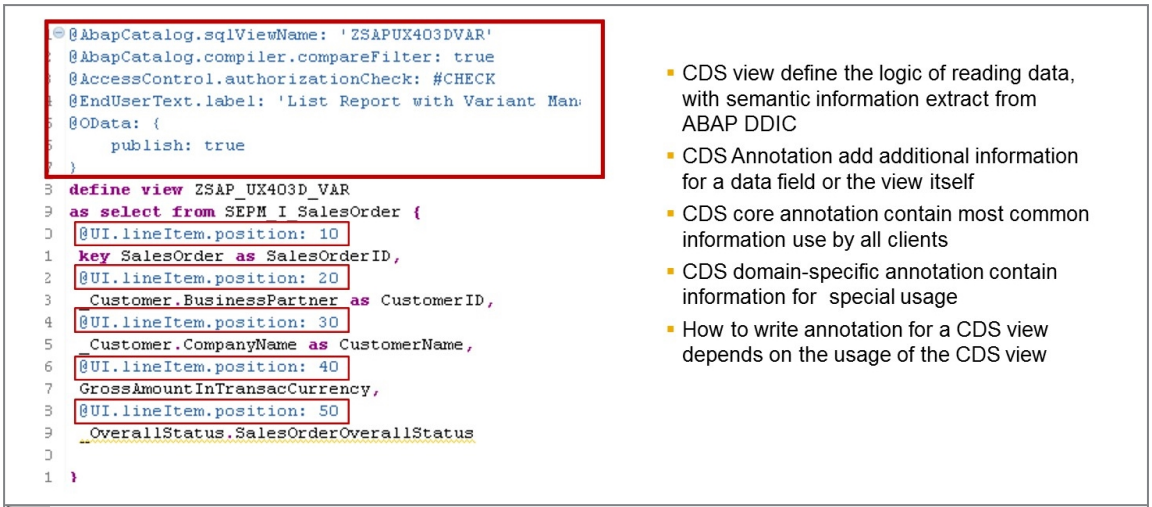


Figure 62: Annotation on CDS

© Copyright. All rights reserved.

CDS annotations are extra information to describe the data, they let the consumer of CDS view know how to use data from the CDS view.

Domain	Usage
UI Annotations	Most important domain for Fiori Elements. Nearly one-to-one relationship between CDS annotation and OData annotation term
Consumption Annotations	Control visibility of a data field or the whole entity Useful when defining value help for a field
Object Model Annotations	Provide definitions of structural as well as transactional related aspects of the business data model
Semantics Annotations	Enrichment of semantic information for data field

Figure 63: Domain Specific Annotation Related to SAP Fiori Elements

Annotations are grouped according to usage, the figure, Domain-Specific Annotation Related to SAP Fiori Elements, shows the most relevant groups of association for SAP Fiori elements.

1. Go to http://help.sap.com
2. Search “ABAP Programming Model for SAP Fiori” and press *Enter*
3. Choose the correct result according your Kernel version
4. Click “Reference” on left bottom of the screen

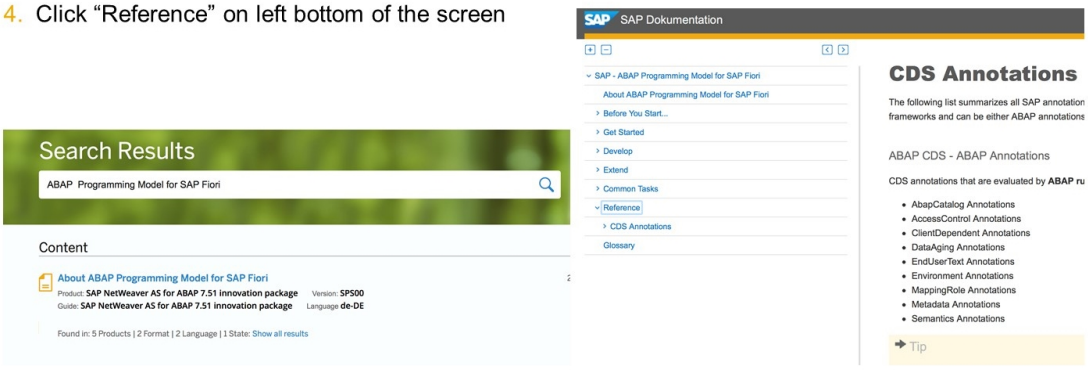


Figure 64: CDS Annotation Reference

To have full knowledge of CDS annotations, you can access the CDS annotation document by searching for **`ABAP Programming Model for SAP Fiori`** on Sap Help Portal.

© Copyright. All rights reserved.

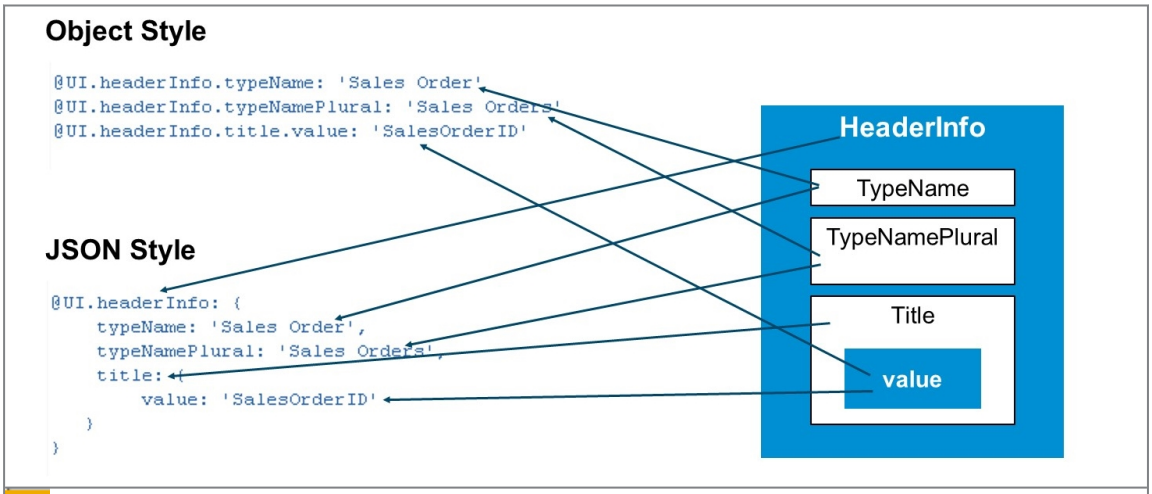


Figure 65: Annotation Syntax: Structure Type

If an annotation term represents a structure, two options are available for writing the annotation.

The object style access properties of annotation, like attributes of a class, use '.' to access sub level properties.

The JSON style access properties like a JSON object. Key feature of this syntax are:

- "{ }" represent a structure
- Use ":" to separate property name and its value
- Use "," to separate properties

Both syntax have same meaning and can be used in combination, such as @UI.headerInfo.title: { value: 'SalesOrderID' }

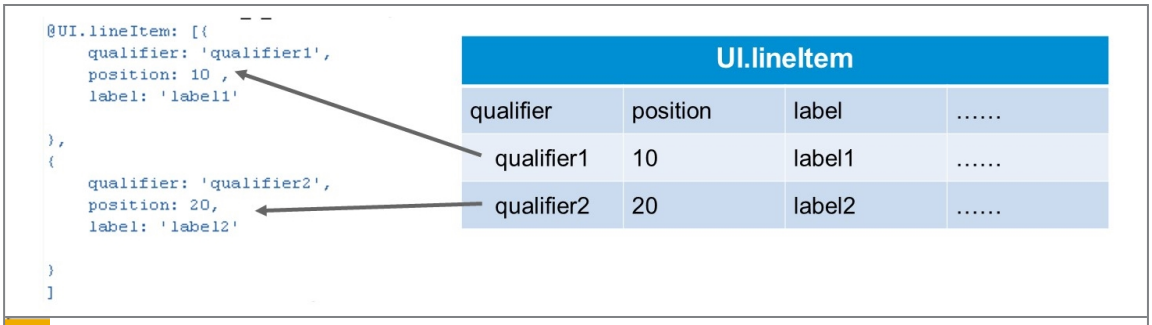


Figure 66: Annotation Syntax: Array Type

If an annotation term represents an array, the only option is to use a JSON array.

Key features of this syntax are:

- "{ }" represent a structure.
- "[ ]" represent an array.
- Use ":" to separate property name and its value.
- Use "," to separate properties.
- In most cases, each line should have a property called "qualifier" to identify it.

© Copyright. All rights reserved.

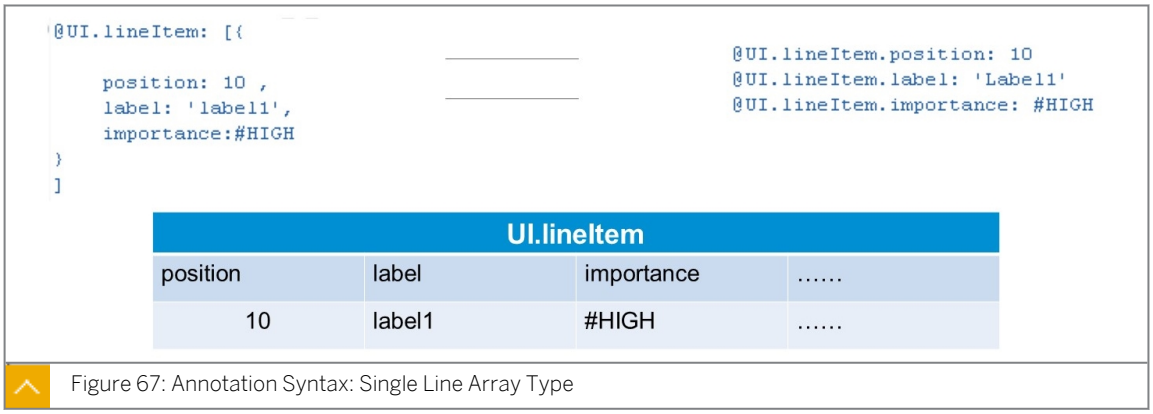


Figure 67: Annotation Syntax: Single Line Array Type

In some cases, a term is an array type, but most of its usage only has single line of record.

In this situation, you can use object style to simplify the input.

CDS Associations

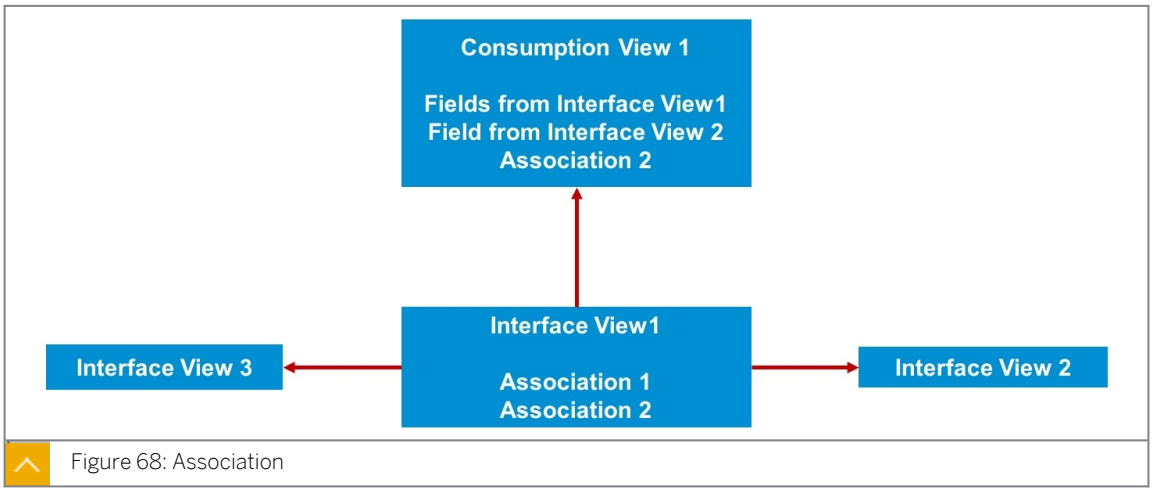


Figure 68: Association

A CDS view can define some associations. An association is a declaration of the relationship of views which can be consumed by itself or by the consumer of this CDS view.

Usually, you need to define some reusable CDS views, which define associations as a representation of the business entity. Then you can write a consumption CDS view , which uses the interface view as its data source, to consume the data and associations.

In the figure Association:

Interface View 1 has two associations connected to Interface View 3 and Interface View 2.

When consumption View 1 accesses data of interface View 1, it can do the following:

- Access fields of Interface View 1..
- Access fields of Interface View 2 through association 1 of interface View 1.
- Expose Association 2 of Interface View 1 again, to let the consumer of this view have the ability to access fields of Interface View 3.

© Copyright. All rights reserved.

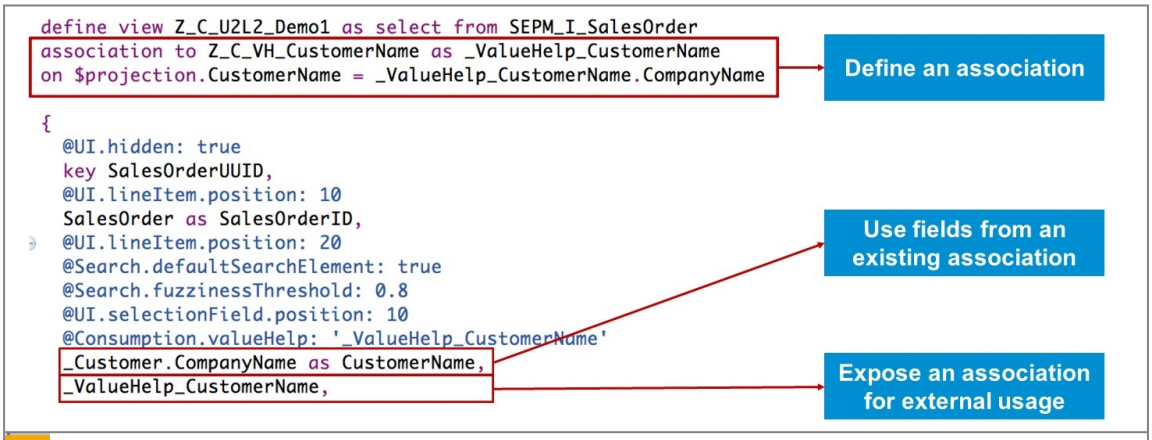


Figure 69: Syntax of Association

The figure, Syntax of association, shows the three main areas of an association.

These are:

Define an association:

The definition of association is quite like SQL JOIN syntax. An easy way to reference fields in the current project list is $projection.

Use fields from an existing association:

You can access fields from associated CDS view by connect association name and fields name with a period '.'. The association can be defined either in current CDS view or the data source view.

Expose an association for external usage:

To expose an association for consumer of this view, write the name of association in the projection list. Since we cannot distinguish association and fields, usually we use '_' as first char for association name.

LESSON SUMMARY

You should now be able to:

- Use the CDS View and SADL

© Copyright. All rights reserved.

Unit 2
Lesson 6

Using the Service Adaption Definition Language (SADL)

LESSON OBJECTIVES
After completing this lesson, you will be able to:

- Use SADL

Service Adaptation Definition Language (SADL)

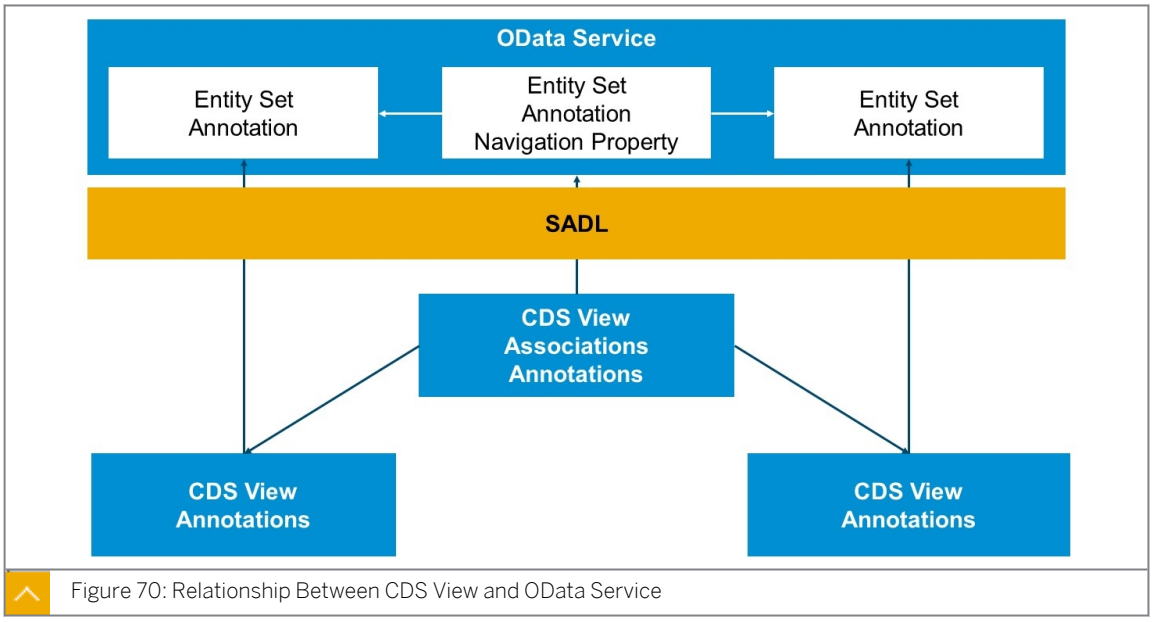


Figure 70: Relationship Between CDS View and OData Service

A framework called SADL can help you publish a group of associated CDS views to an OData service. Each CDS view is an Entity Set, and their associations are translated to OData associations. The association name reflects in the OData entity as a Navigation Property.

All SAP Fiori elements related annotations are translated to OData annotations. In this way, you can combine your business logic, OData service development, and annotation into one CDS view.

© Copyright. All rights reserved.

CDS as an OData Service

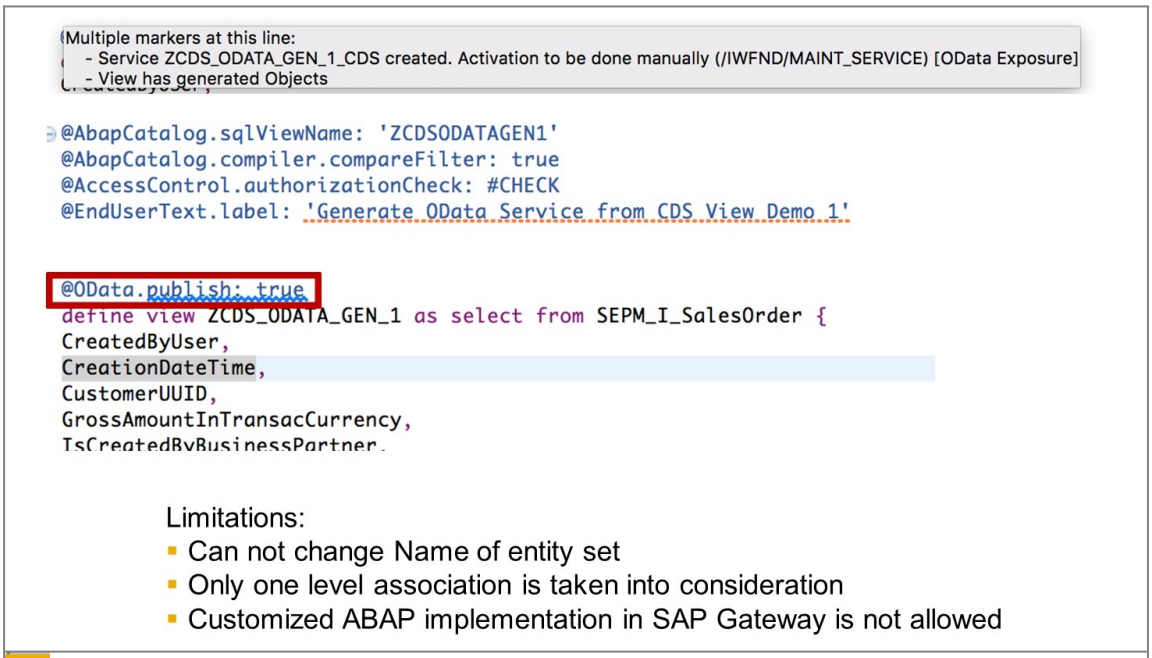


Figure 71: Expose CDS as an OData Service

If a CDS view has the annotation "@Odata.Publish:true", an OData service is generated automatically in your back end system. The only thing you need to do is to activate it in your front end system.

The CDS view you are working on is transformed to an EntitySet with a name identical to the generated OData service.

All CDS views associated to this view are also transformed to an EntitySet. The association relationship is transformed to the Association and Navigation Property in the OData service.

Creating an OData Service by Referencing a CDS

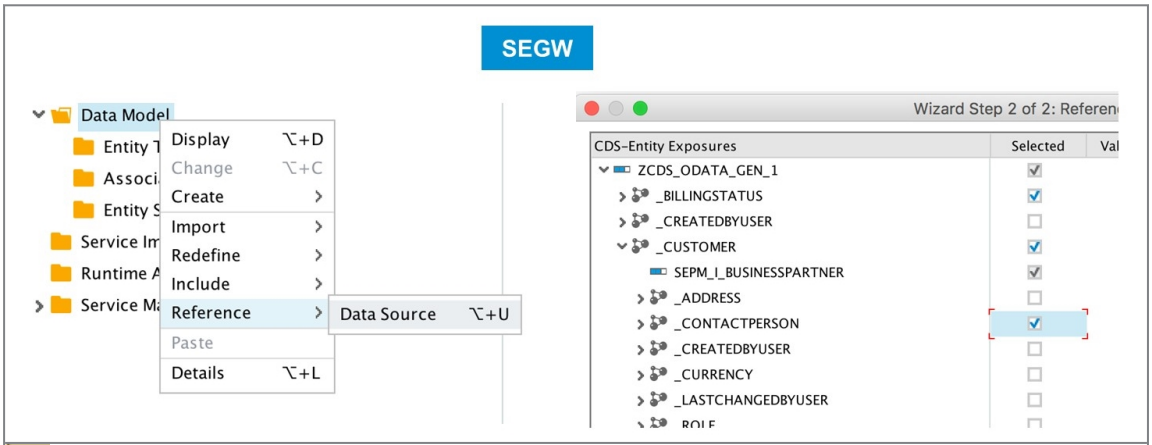


Figure 72: Create OData service by reference a CDS

In SAP Gateway Service Builder, you can reference an existing CDS view. Then, you can add associated CDS views selectively, even associations of associated views can be added to your OData Service.

© Copyright. All rights reserved.

You can then override MPC or DPC standard implementation to add write support and advanced OData features for the OData Service.

> Note:
> For detailed information of how to create an OData service by reference to a CDS, please refer to GW100.

Relationship Between CDS Annotation and OData Annotation

Figure 73: Relationship Between CDS Annotation and OData Annotation

Like OData Annotations, CDS Annotations can apply on an EntitySet or a data field.

In most cases, there is a 1:1 relationship between an OData annotation term and a CDS annotation, with a similar name.

CDS Annotations are grouped by target. Annotations with the same target stay together.

OData Annotations do not have a fixed position. It is usually grouped by terms.

When translated to an OData annotation, the sequence of CDS annotations belonging to one term is determined by a position property.

LESSON SUMMARY

You should now be able to:

- Use SADL

© Copyright. All rights reserved.

© Copyright. All rights reserved.

Explaining Metadata Extension

LESSON OBJECTIVES
After completing this lesson, you will be able to:

- Explain Metadata Extension

Metadata Extension

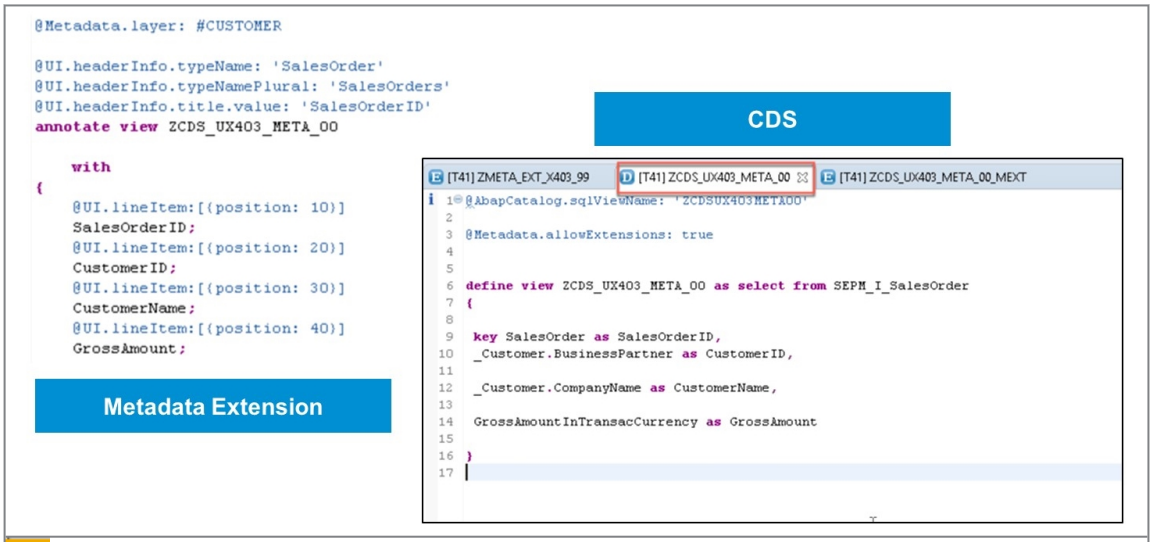


Figure 74: Metadata Extension

The syntax of CDS will grow if complex annotations are needed, that may reduce the readability of CDS.

A metadata extension separate annotation from business logic.

To implement a metadata extension, you need to do the following:

1. Add an annotation of *@metadata.allExtensions:true* to the CDS view.
2. Create a Metadata extension, write annotations for the view and its fields.
3. Use ";" to separate fields.

Note:
Metadata extensions only allow annotation of JSON style.

Some annotations relevant to the activation of CDS, like *@odata.publish, @AbapCatalog.sqlViewName* are not supported in a metadata extension.

© Copyright. All rights reserved.

Note:
If there is more than one metadata extension annotation for the same CDS, please reference: https://help.sap.com/doc/abapdocu_752_index_htm/7.52/en-US/abencds_meta_data_extension_eval.htm for how to determine which metadata extension is used.

LESSON SUMMARY
You should now be able to:

- Explain Metadata Extension

© Copyright. All rights reserved.

Unit 2
Lesson 8

Learning Scenarios of Fiori Elements Implementation

LESSON OBJECTIVES
After completing this lesson, you will be able to:

- Learn about scenarios of Fiori Elements implementation

Scenarios

System Requirements for SAP Fiori Elements Development with SADL

	NetWeaver	Web IDE	Related S/4HANA Version
List Report	>= 7.50 SP01	>= 1.17	S/4HANA 1511
Object Page	>= 7.50 SP01	>=1.17	S/4HANA 1511
Overview Page	>= 7.51	>= 1.17	S/4HANA 1511
Analytic List Page	>= 7.52	>= 1.17	S/4HANA 1709

* NetWeaver 7.50 support only few annotations, most advanced annotations are supported in 7.51
* Some features, like analytical report, are only work with HANA database

Figure 75: System Requirements for SAP Fiori Elements Development with SADL

To use SAP Fiori elements in SAP NetWeaver AS ABAP, the minimum version of SAP NetWeaver is 7.50 SP01, otherwise the SADL will not work.

Lots of new annotations are added in each upgrade of SAP NetWeaver, SAP Fiori elements will be more powerful if you choose the newest version of SAP NetWeaver.

In general, CDS works on any database, but SAP HANA is recommended when good performance is needed, due to CDS push down the calculation to the DB layer.

© Copyright. All rights reserved.

System Requirements for SAP Fiori Elements Development with Local Annotation

	SAPUI5/OpenUI5	SAP_UI
List Report	>1.38	SAP_UI 7.50 and UI add-on 2.0
Object Page	>1.38	SAP_UI 7.50 and UI add-on 2.0
Overview Page	>1.44	SAP_UI 7.51
Analytic List Page	>1.48	SAP_UI 7.52

Figure 76: System Requirements for SAP Fiori Elements Development with Local Annotation

When writing annotation as a local file using SAP WebIDE, SAP Fiori elements works with any OData service, even from non-SAP back end.

If SAPUI5 runs on non-ABAP platform, like SAP CP, or you're using OpenUI5, SAP Fiori elements only requires the version of SAPUI5/OpenUI5.

If SAPUI5 runs on ABAP platform, SAP Fiori elements require version of SAP_UI.

SADL vs Local Annotation

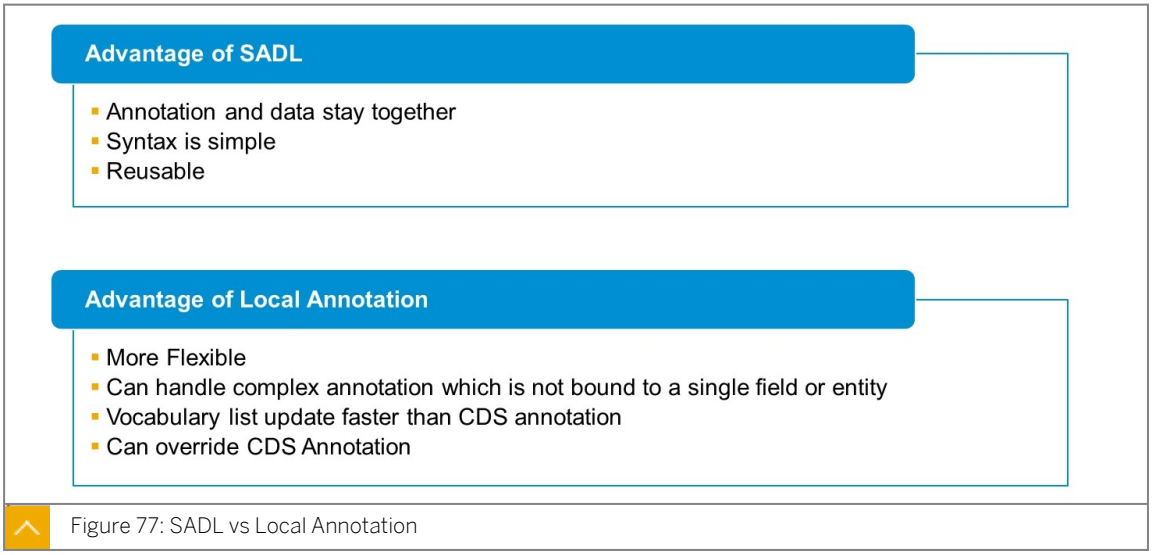


Figure 77: SADL vs Local Annotation

In most cases, UI annotations in CDS view can be translated to OData annotation by SADL framework, but they are not 100% the same and have their own advantages.

© Copyright. All rights reserved.

Suggestions to Choose the Correct Method of Annotation

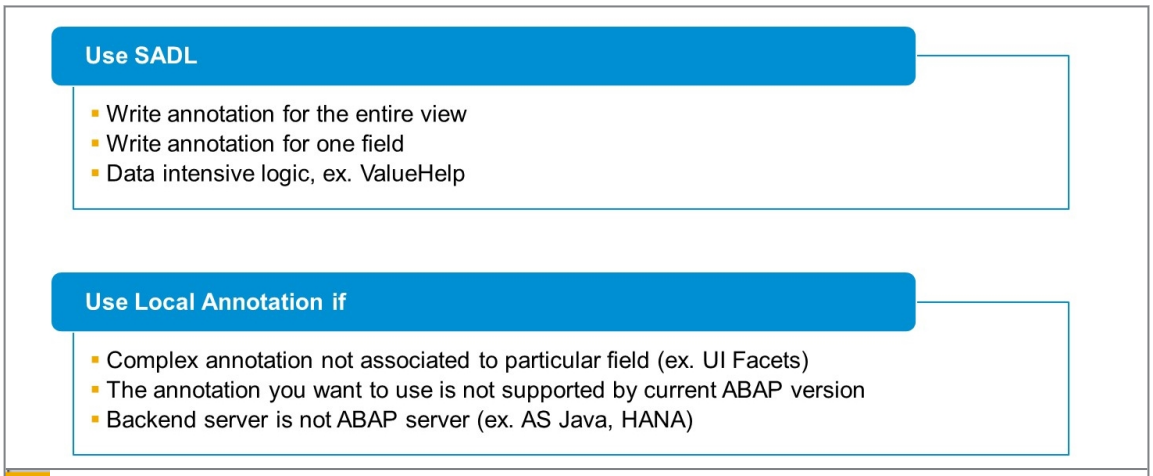


Figure 78: How to Choose Methods of Annotation

The figure, How to Choose Methods of Annotation, lists some factors for choosing the methods of annotation.

LESSON SUMMARY

You should now be able to:

- Learn about scenarios of Fiori Elements implementation

© Copyright. All rights reserved.

© Copyright. All rights reserved.

Learning Assessment

1. What are the features of SAP Fiori elements?

   Choose the correct answers.

   - ☐ **A** No JavaScript UI Coding.
   - ☐ **B** Metadata-driven approach of SAP Fiori development.
   - ☐ **C** A replacement for traditional free style SAPUI5 programming, can satisfy all customer needs in a brand new approach.
   - ☐ **D** Centrally Provided Templates covering Reporting, Analytic, and Transaction scenarios.

2. Which of the following information about an OData service should be provided as an annotation?

   Choose the correct answer.

   - ☐ **A** The entities of an OData service
   - ☐ **B** Properties of an Entity Set/Collection
   - ☐ **C** The position for each field in a list report
   - ☐ **D** Data type for each property in an Entity Set/ Connection

3. The combination of Term/Target is unique, that means for an Entity or a field, every term can be used only once.

   Determine whether this statement is true or false.

   - ☐ True
   - ☐ False

© Copyright. All rights reserved.

4. Which of following templates display only one business entity?

   Choose the correct answer.

   - ☐ A List Report
   - ☐ B Object Page
   - ☐ C Overview Page
   - ☐ D Analytic List Page

5. Using SAP S/4HANA 1610 with ABAP 7.51, which of following templates can you use for SAP Fiori elements?

   Choose the correct answers.

   - ☐ A List Report
   - ☐ B Object Page
   - ☐ C Overview Page
   - ☐ D Analytic List Page

6. Which info will be used for creating a destination in SAP BTP for SAP Fiori elements development?

   Choose the correct answer.

   - ☐ A Virtual name in SAP Cloud Connector
   - ☐ B Internal name in SAP Could Connector
   - ☐ C External name in SAP Cloud Connector
   - ☐ D Internal address of SAP Backend Server

7. When creating a CDS view, the SQL view name and view name for CDS must be identical.

   Determine whether this statement is true or false.

   - ☐ True
   - ☐ False

© Copyright. All rights reserved.

8. Which of following steps are needed to create an association in CDS and expose it?

   Choose the correct answers.

   - ☐ A Declare an association using “association to” statement.
   - ☐ B Declare an association using “left outer join” statement.
   - ☐ C Expose the association by writing its name in projection list.
   - ☐ D Expose fields in the association by writing each field in projection list.

9. What are the limitations of publishing CDS as OData service by adding a *@OData.publish:true*?

   Choose the correct answers.

   - ☐ A Cannot expose associations.
   - ☐ B Cannot change names of entity sets.
   - ☐ C Only 1 level is taken into consideration when exposing associations.
   - ☐ D No customized ABAP code in SAP Gateway.

10. In which cases, is a local annotation better than a CDS annotation?

    Choose the correct answers.

    - ☐ A UI with data intensive.
    - ☐ B Annotations is for 1 field.
    - ☐ C Complex UI relevant annotations.
    - ☐ D You want to use annotations which are not support by your current ABAP version.

© Copyright. All rights reserved.

Learning Assessment - Answers

1. What are the features of SAP Fiori elements?

   Choose the correct answers.

   - [X] A No JavaScript UI Coding.
   - [X] B Metadata-driven approach of SAP Fiori development.
   - [ ] C A replacement for traditional free style SAPUI5 programming, can satisfy all customer needs in a brand new approach.
   - [X] D Centrally Provided Templates covering Reporting, Analytic, and Transaction scenarios.

   Correct. No JavaScript UI coding, metadata-driven approach of SAP Fiori development, and centrally provided templates covering reporting, analytic, and transaction scenarios are the features of SAP Fiori elements.

2. Which of the following information about an OData service should be provided as an annotation?

   Choose the correct answer.

   - [ ] A The entities of an OData service
   - [ ] B Properties of an Entity Set/Collection
   - [X] C The position for each field in a list report
   - [ ] D Data type for each property in an Entity Set/ Connection

   Correct. The position for each field in a list report should be provided as an annotation in the OData service.

3. The combination of Term/Target is unique, that means for an Entity or a field, every term can be used only once.

   Determine whether this statement is true or false.

   - [ ] True
   - [X] False

   Correct. The combination of Term/Target is not unique.

© Copyright. All rights reserved.

4. Which of following templates display only one business entity?

   Choose the correct answer.

   - [ ] A List Report
   - [x] B Object Page
   - [ ] C Overview Page
   - [ ] D Analytic List Page

   Correct. An Object Page displays only one business entity.

5. Using SAP S/4HANA 1610 with ABAP 7.51, which of following templates can you use for SAP Fiori elements?

   Choose the correct answers.

   - [x] A List Report
   - [x] B Object Page
   - [x] C Overview Page
   - [ ] D Analytic List Page

   Correct. List Report, Object Page, and Overview Page, can be used for SAP Fiori elements.

6. Which info will be used for creating a destination in SAP BTP for SAP Fiori elements development?

   Choose the correct answer.

   - [x] A Virtual name in SAP Cloud Connector
   - [ ] B Internal name in SAP Could Connector
   - [ ] C External name in SAP Cloud Connector
   - [ ] D Internal address of SAP Backend Server

   Correct. Virtual name in SAP BTP will be used for creating a destination in SAP BTP for SAP Fiori elements development.

7. When creating a CDS view, the SQL view name and view name for CDS must be identical.

   Determine whether this statement is true or false.

   - [ ] True
   - [x] False

   Correct. When creating a CDS view, the SQL view name and view name for CDS must not be identical.

© Copyright. All rights reserved.

8. Which of following steps are needed to create an association in CDS and expose it?

   Choose the correct answers.

   - [X] A Declare an association using “association to” statement.
   - [ ] B Declare an association using “left outer join” statement.
   - [X] C Expose the association by writing its name in projection list.
   - [ ] D Expose fields in the association by writing each field in projection list.

   Correct. Declare an association using “association to” statement and expose the association by writing its name in projection list are the required steps needed to create an association in CDS and expose it.

9. What are the limitations of publishing CDS as OData service by adding a *@OData.publish:true*?

   Choose the correct answers.

   - [ ] A Cannot expose associations.
   - [X] B Cannot change names of entity sets.
   - [X] C Only 1 level is taken into consideration when exposing associations.
   - [X] D No customized ABAP code in SAP Gateway.

   Correct. Cannot change names of entity sets. Only 1 level is taken into consideration when exposing associations and no customized ABAP code in SAP Gateway are the limitations.

10. In which cases, is a local annotation better than a CDS annotation?

    Choose the correct answers.

    - [ ] A UI with data intensive.
    - [ ] B Annotations is for 1 field.
    - [X] C Complex UI relevant annotations.
    - [X] D You want to use annotations which are not support by your current ABAP version.

    Correct. Complex UI relevant annotations and you want to use annotations which are not support by your current ABAP version.

© Copyright. All rights reserved.

UNIT 3 List Report

UNIT OBJECTIVES

- Explain basic annotations for creating a list report
- Describe options for adjusting the display of columns
- Use Searching and Filtering data
- Provide the value help
- Explain Variant Management

© Copyright. All rights reserved.

© Copyright. All rights reserved.

Explaining Basic Annotations for List Report

LESSON OBJECTIVES

After completing this lesson, you will be able to:

- Explain basic annotations for creating a list report
- Describe options for adjusting the display of columns

Basic Annotations for Creating a List Report

Mandatory Annotations

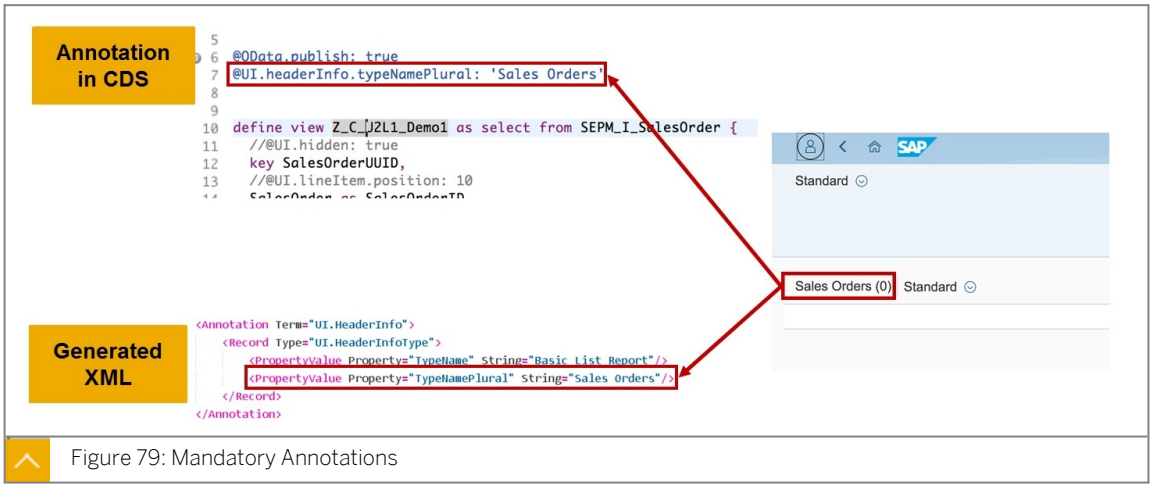


Figure 79: Mandatory Annotations

The following are typical mandatory annotations:

@OData.publish:true

Auto exposures the CDS view as OData Service.

@UI.headerInfo.typeNamePlural

Determines the text displayed on the upper left corner of the list report.

© Copyright. All rights reserved.

Adding Columns to Report

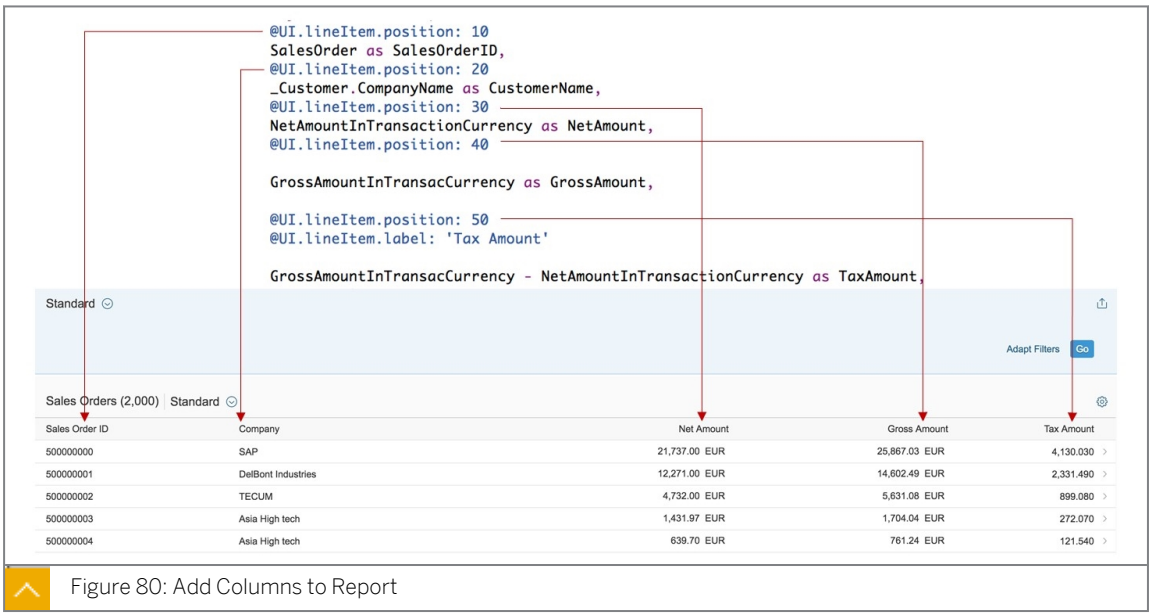


Figure 80: Add Columns to Report

Annotation *@UI.lineItem* is used to set default columns for list report.

The Data types for *@UI.lineItem* are a collection of Data Fields, but in most cases only one *@UI.lineItem* exists for a field.

The mandatory property is **position**, which determines the sequence of columns. The label for each field comes from field label of data element for a fields.

You can also add a calculated field to a list report. Since the fields do not reference a data element, you need to either assign it a label by using *@UI.lineItem.lable*, or convert the data type to a data element by using function *cast*.

Add Columns to with Guided Development

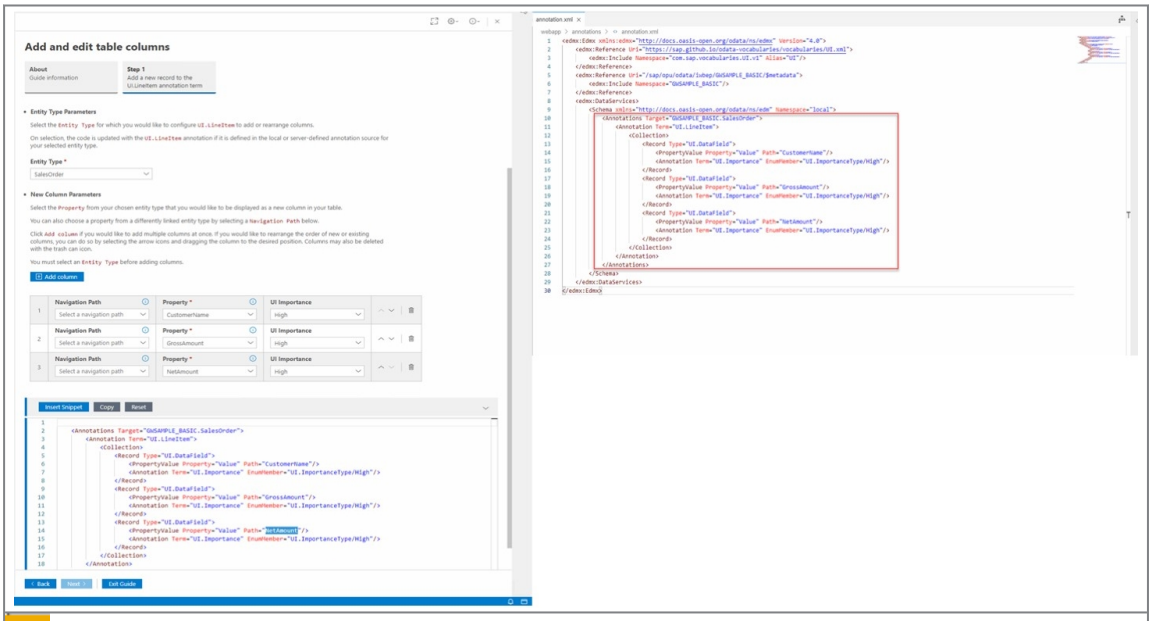


Figure 81: Add Columns

© Copyright. All rights reserved.

Semantic Information for Amount/Quantity fields

Figure 82: Add Semantic Information for Amount/Quantity fields

The following items should be noted when adding semantic Information for Amount/Quantity fields:

- For amount and quantity fields, you should point out where to find its currency/unit fields.
- Fields selecting from DDIC, or other CDS entities may already have this information. If it is defined correctly, the currency/unit displays with the amount/quantity fields.

If it's not displayed correctly, you should add semantic information in your CDS view, using the following:

- @semantics.currencycode/@semantics.unitofmeasure to annotate for currency/unit field
- @semantics.amount.currencycode/@semantics.quantity.unitofmeasure to annotate amount/quantity field and tell them which field has the information of currencyCode/ unit of measure

You should always annotate semantics information for calculated fields that you declared in your CDS view.

© Copyright. All rights reserved.

Options for Adjusting the Display of Columns

Importance of Columns

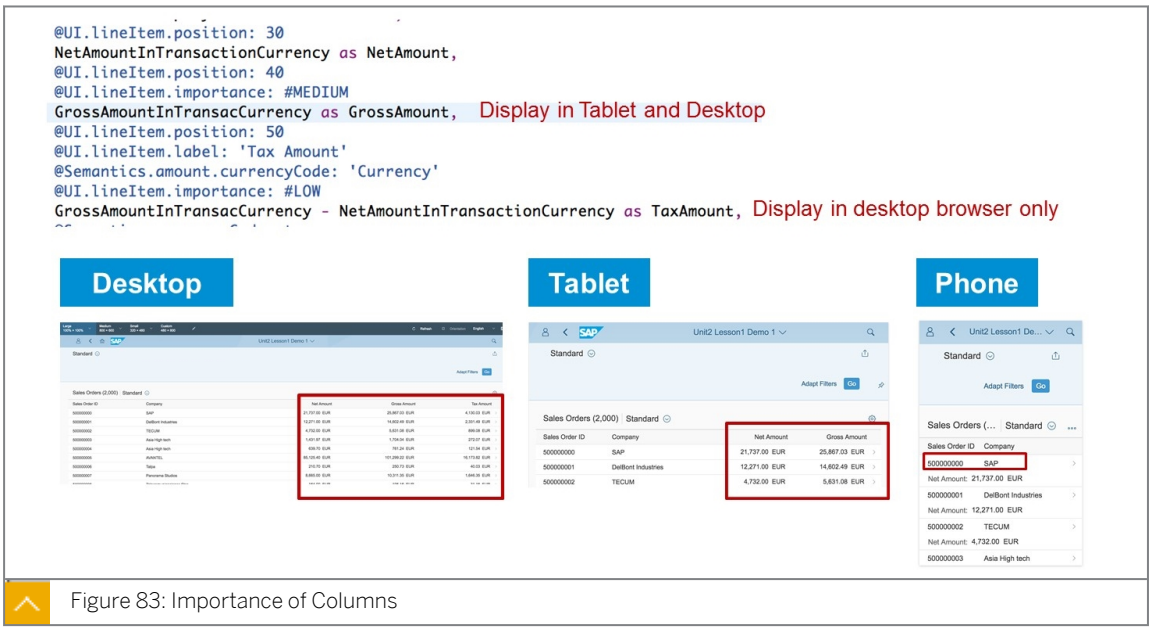


Figure 83: Importance of Columns

The controls used in a list report are designed for all clients.

Different clients have different screen widths. The fields displayed in a Smart Phone should be smaller than a desktop browser.

Use `@UI.lineitem.importance` to determine in which clients the field should display:

- #HIGH: Default value, display in all clients
- #MEDIUM: Only display in desktop browser or tablet
- #LOW: Only display in desktop browser

© Copyright. All rights reserved.

Importance of Columns Guided Development

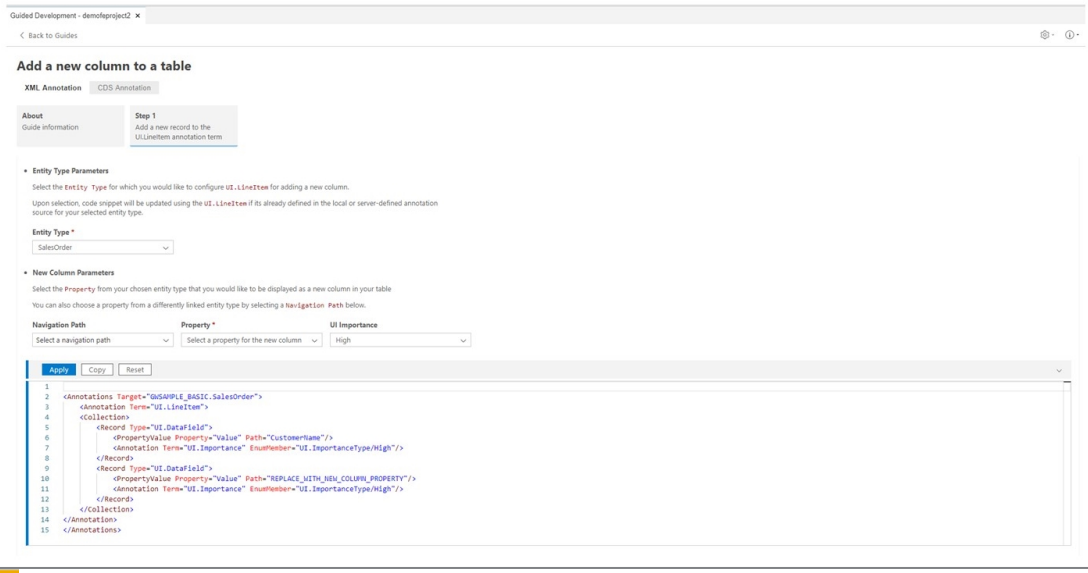


Figure 84: Importance of Columns Guided Development

The OData annotation generated in XML format looks like the data in the upper part of the figure.

Hiding Fields

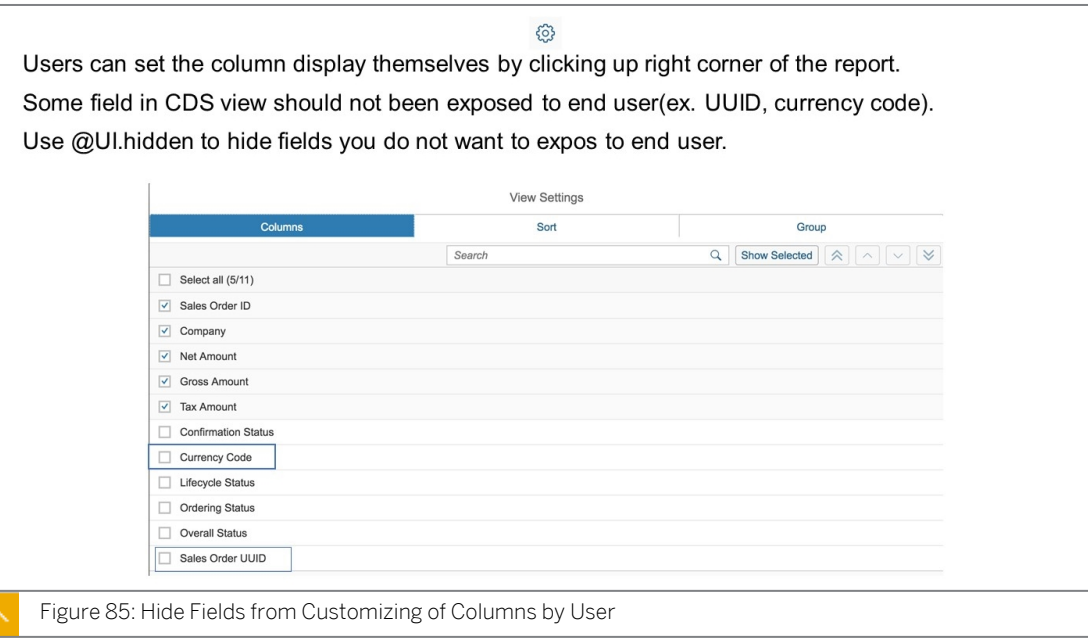


Figure 85: Hide Fields from Customizing of Columns by User

Some fields are not suitable for displaying on the report. You can use `@UI.hidden` to hide them in the table settings dialog.

LESSON SUMMARY

You should now be able to:

- Explain basic annotations for creating a list report

© Copyright. All rights reserved.

- Describe options for adjusting the display of columns

© Copyright. All rights reserved.

Using Searching and Filtering Data

LESSON OBJECTIVES

After completing this lesson, you will be able to:

- Use Searching and Filtering data

Searching and Filtering Data

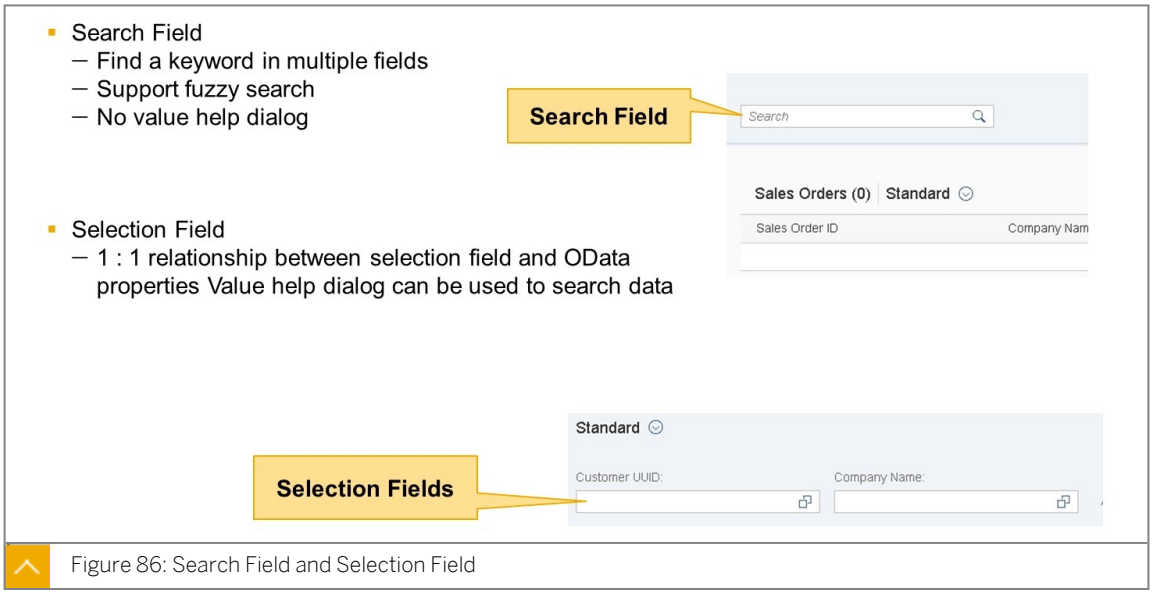


Figure 86: Search Field and Selection Field

Searching and filtering data are fundamental functions for a report.

The List Report supports two ways of search and filtering:

- **Search Field** is a field in the top left corner of a list report. Users can search data by entering a keyword in the search field. The report searches data across several fields according to the rule defined by developer.
- **Selection Fields** are input fields with value help dialogs on top of a list report. Each selection fields represent a filter for one property of the back end OData service. Users can customize selection fields of their own and save them as a variant.

© Copyright. All rights reserved.

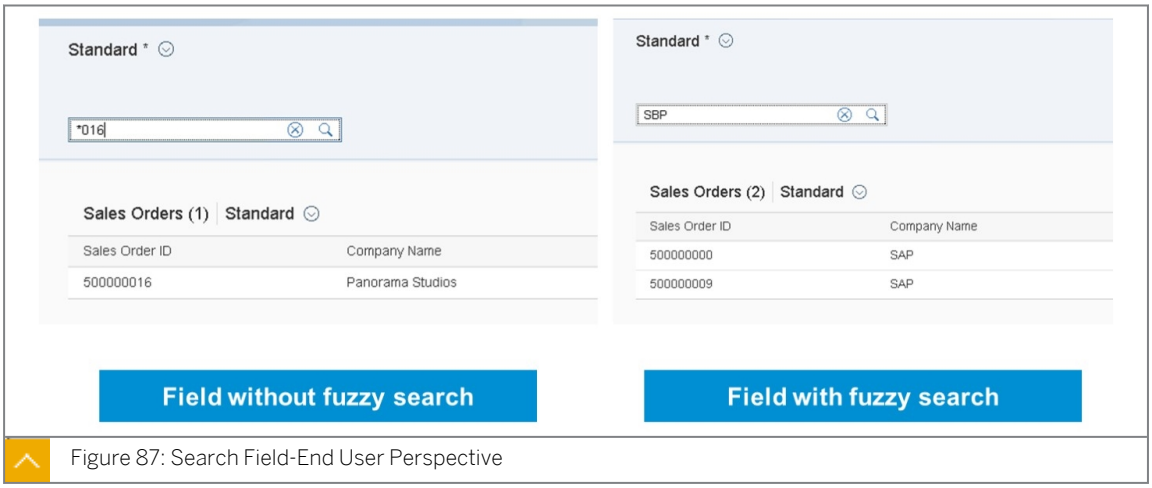


Figure 87: Search Field-End User Perspective

For fields without a fuzzy search function, users must input the exactly same keyword as the data, or use wildcard characters (* and ?) to match data.

For fields with a fuzzy search function, if a user enters a typo in their keyword, similar results appear. In this example, the keyword is 'SBP', but SAP appears in the result.

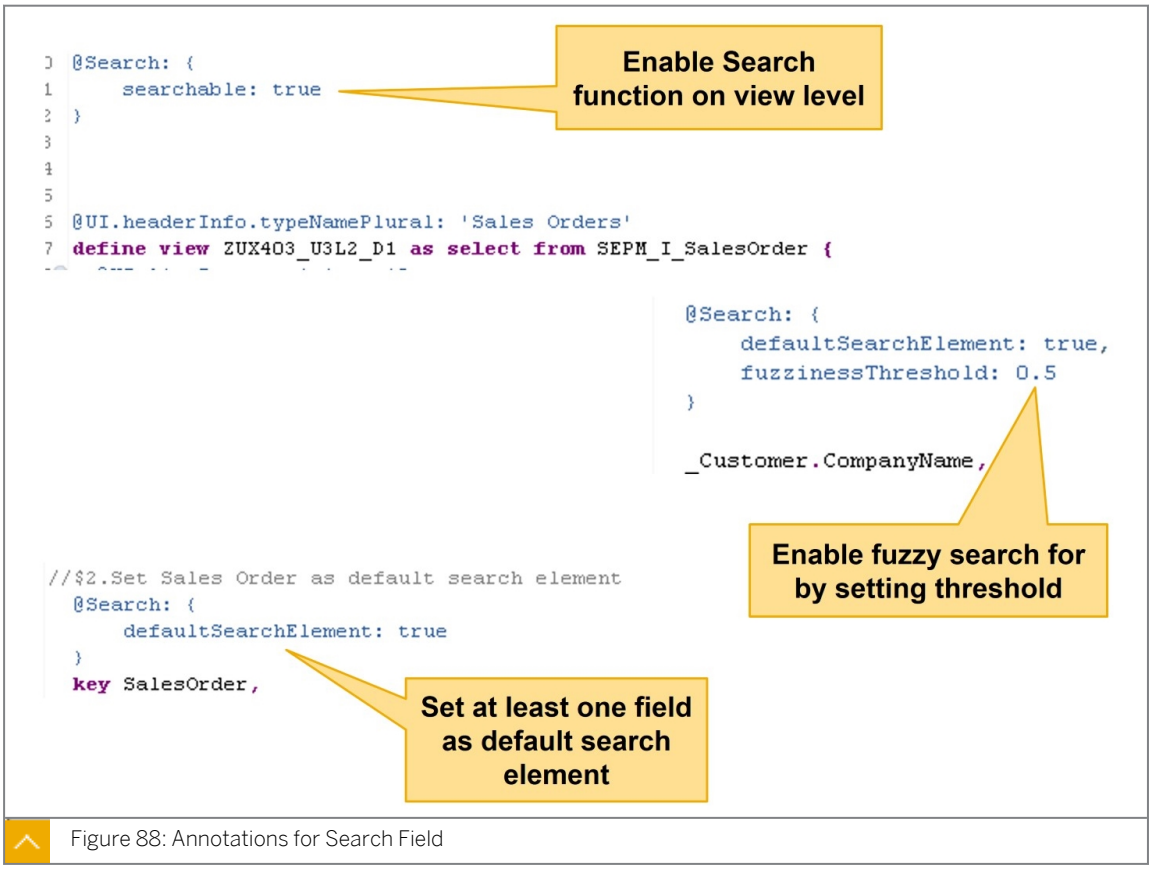


Figure 88: Annotations for Search Field

To add a search field for a list report, a developer must unable it on CDS view level by adding a *@Search.searchable:true* annotation.

At least one field should be set as default search element. If a field needs a fuzzy search function, a threshold value must be set.

The threshold value is a number between 0 and 1. The search result will be more accurate and return less records when the value increases, and vice versa.

© Copyright. All rights reserved.

Generally, a value in common usage is 0.8.

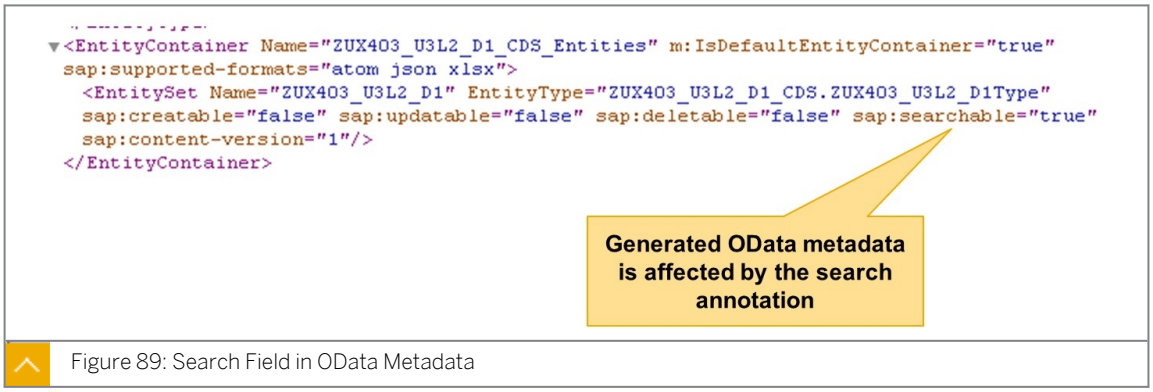


Figure 89: Search Field in OData Metadata

Unlike other annotations, the annotations for search fields only affect metadata of OData service.

There is an OData V2 annotation on the entity set generated by the CDS view.

Default search elements and fuzzy search related annotations are not relevant with UI, so they do not appear in OData annotation.

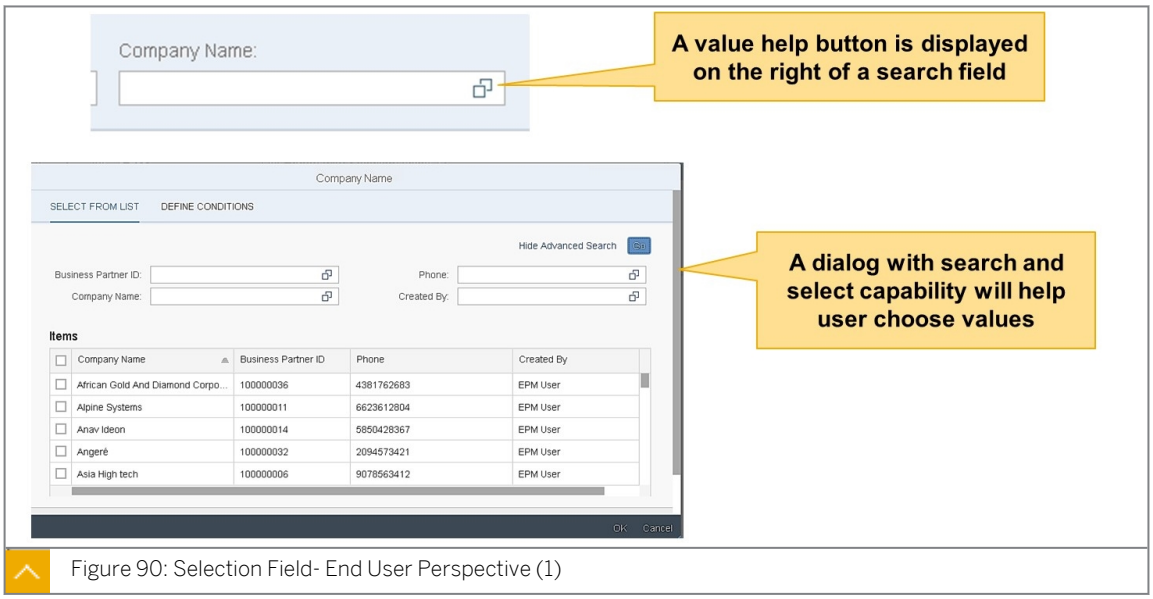


Figure 90: Selection Field- End User Perspective (1)

For each search field, a value help button is displayed on the right.

After the user chooses the button, a dialog with search and select capability displays to help the user choose values.

It is a developer's responsibility to provide a value help dialog for the end user.

© Copyright. All rights reserved.

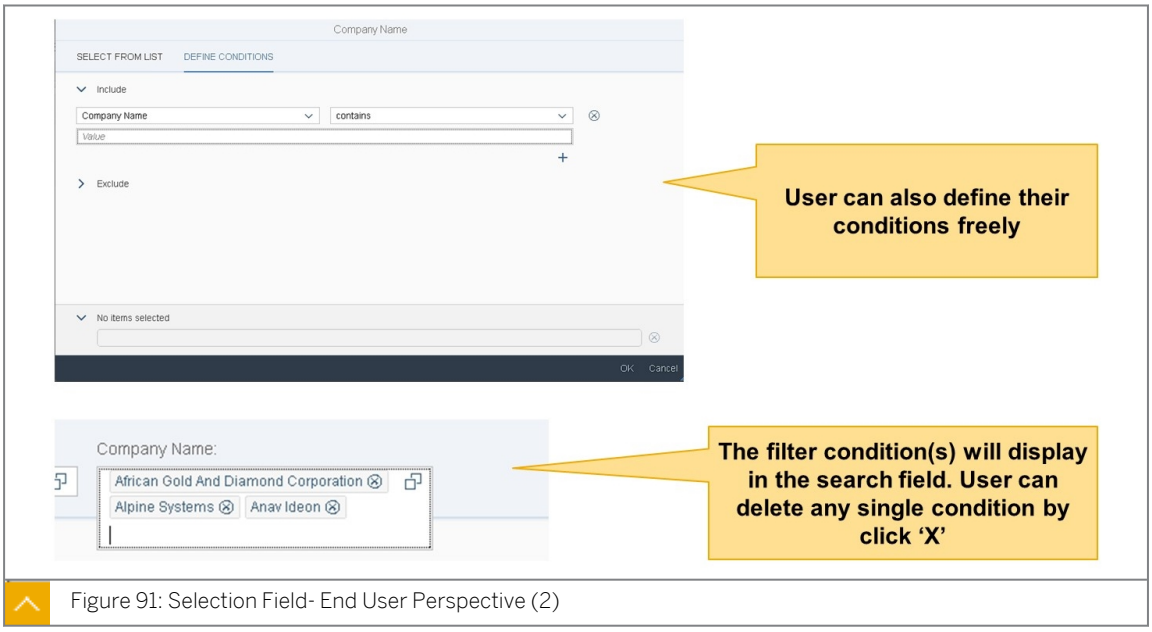


Figure 91: Selection Field- End User Perspective (2)

If a chosen value from a list is not in the casing that the user wanted, the entry also supports free style input of conditions.

If the developer has not provided a value help dialog, only free style input is supported.

When the condition input is complete, all conditions display in the search field. Users can delete any single condition by choosing *X* on the right.

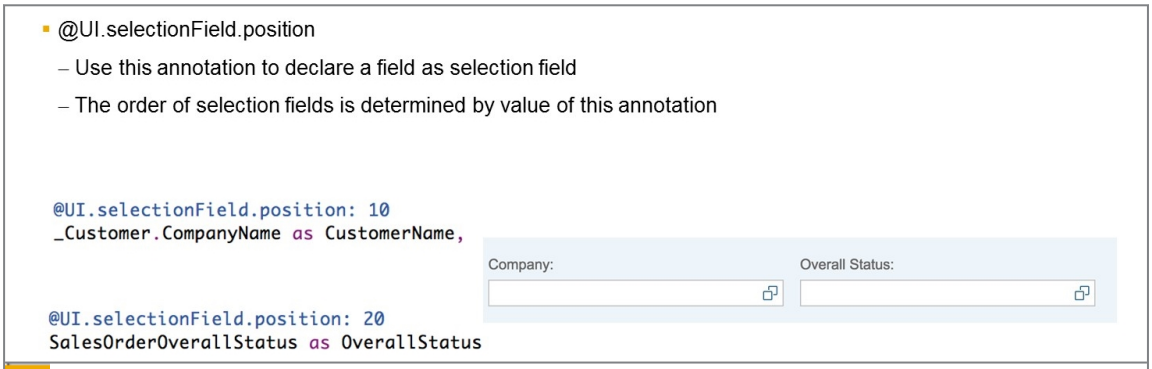


Figure 92: Create Selection Field for CDS Elements

To enable a selection field, use `@UI.selectionField.position` as element annotation on the field that needs to be a filter.

Even fields that do not display as a column can be used as a filter condition.

© Copyright. All rights reserved.

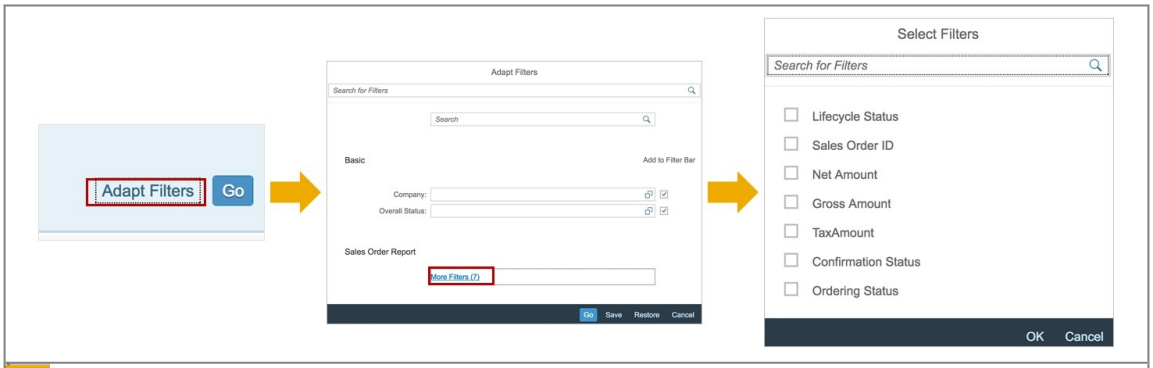


Figure 93: Adapt Filters

However, at runtime, end users can choose any fields as a filter by choosing *Adapt Filters*.

LESSON SUMMARY

You should now be able to:

- Use Searching and Filtering data

© Copyright. All rights reserved.

© Copyright. All rights reserved.

Providing the Value Help

LESSON OBJECTIVES

After completing this lesson, you will be able to:

- Provide the value help

Value Help

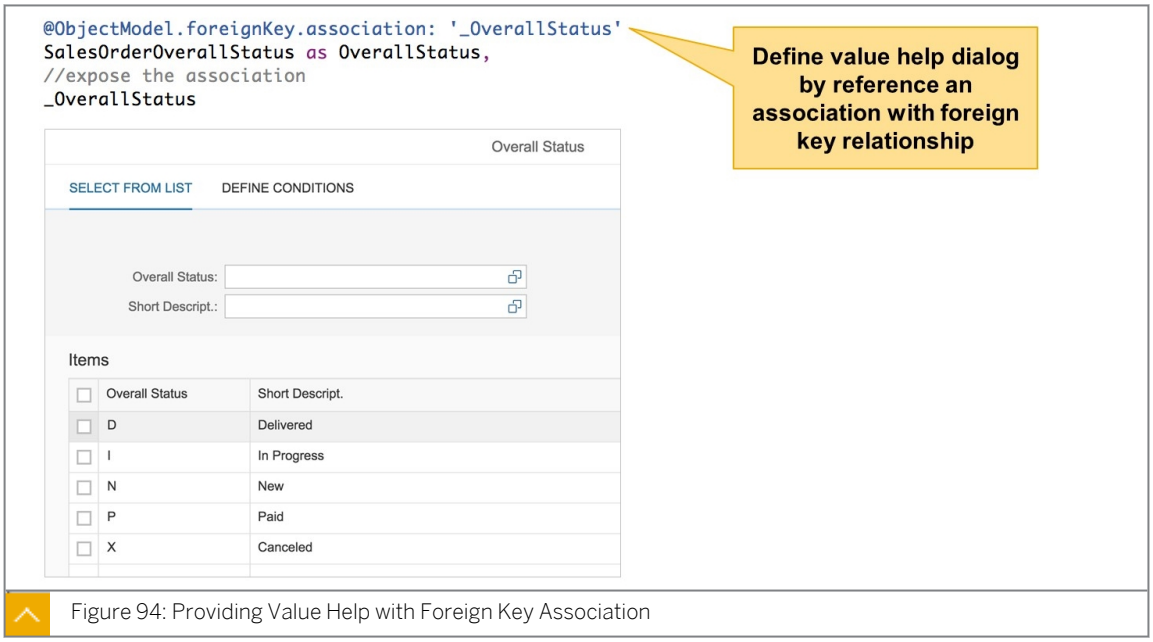


Figure 94: Providing Value Help with Foreign Key Association

If there is an association for a field, you can use this association as a check table for the field. Then, you can provide a value help for the user. The following steps are used to provide value help with foreign key association:

1. Find appropriate association for value help.
2. Annotate the selection field with *@ObjectModel.foreignkey.association*.
3. Expose the association.

© Copyright. All rights reserved.

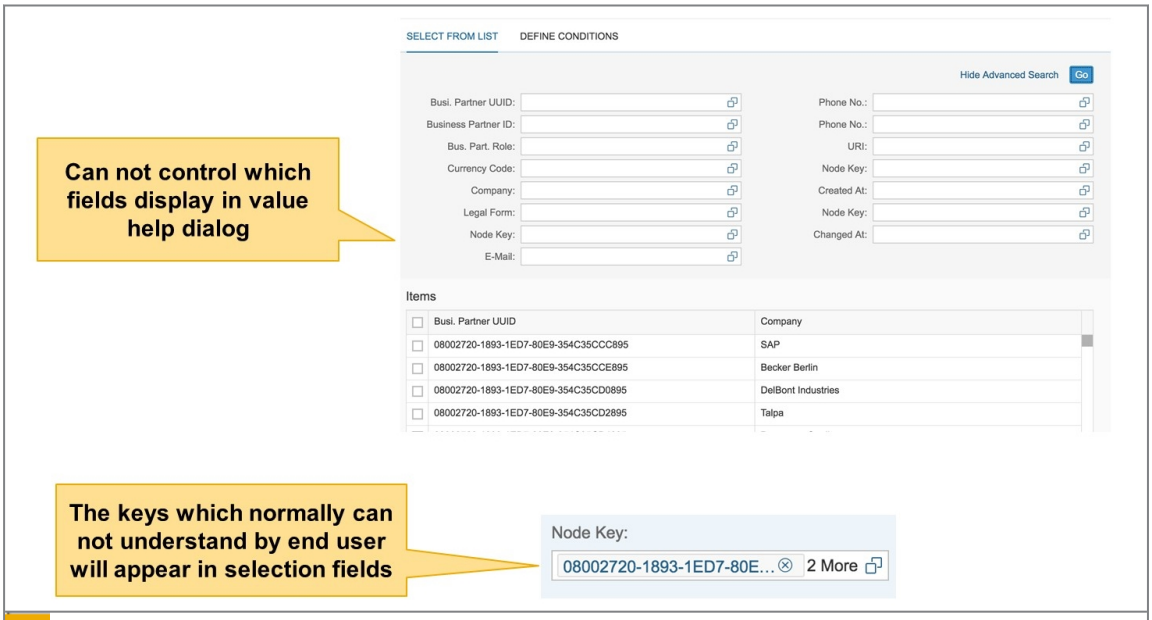


Figure 95: Limitations of Foreign Key Based Value Help

Although it's easy to declare a foreign key based value help, it has the following limitations:

- Normally, only technical key fields of a business entity have the association to other views, which are fields we want to hide from the end user.
- In the search help window, only the key field and one field annotated with *@semantics.text:true* can be displayed in the search result window.
- All fields from check view are present as selection fields of the value help screen.

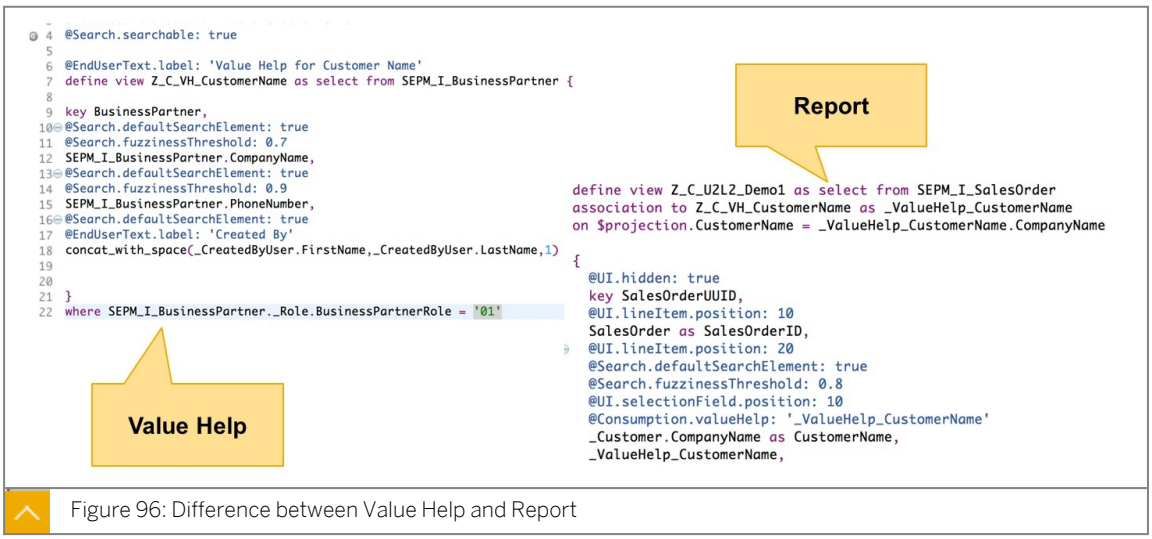


Figure 96: Difference between Value Help and Report

For a better user experience, use the following steps to create a modeled value help view:

1. Create a CDS view, select data from existing table or CDS view. Expose only the fields you want to display in value help.
2. You can add additional where conditions and calculated fields for your value help view.
3. Create an association from your report CDS view to the value help view, associate them with a selection field, which is understandable by end users.

© Copyright. All rights reserved.

4. Annotate the selection field with `@Comsuption.valuehelp:'<association_name>`.
5. Expose the association.
6. You can add full text fuzzy search function for your value help by adding `@search` annotation in your value help CDS view.

Note:

- Do not use calculated fields as search conditions if they are not required, it may lead to performance issues.
- For calculated fields in value help view, use `@EndUserText.label` to set a label.
- Redeploy your report CDS view after you changed and activated the value help view.

LESSON SUMMARY

You should now be able to:

- Provide the value help

© Copyright. All rights reserved.

© Copyright. All rights reserved.

Explaining Variant Management

LESSON OBJECTIVES
After completing this lesson, you will be able to:

- Explain Variant Management

Variant Management

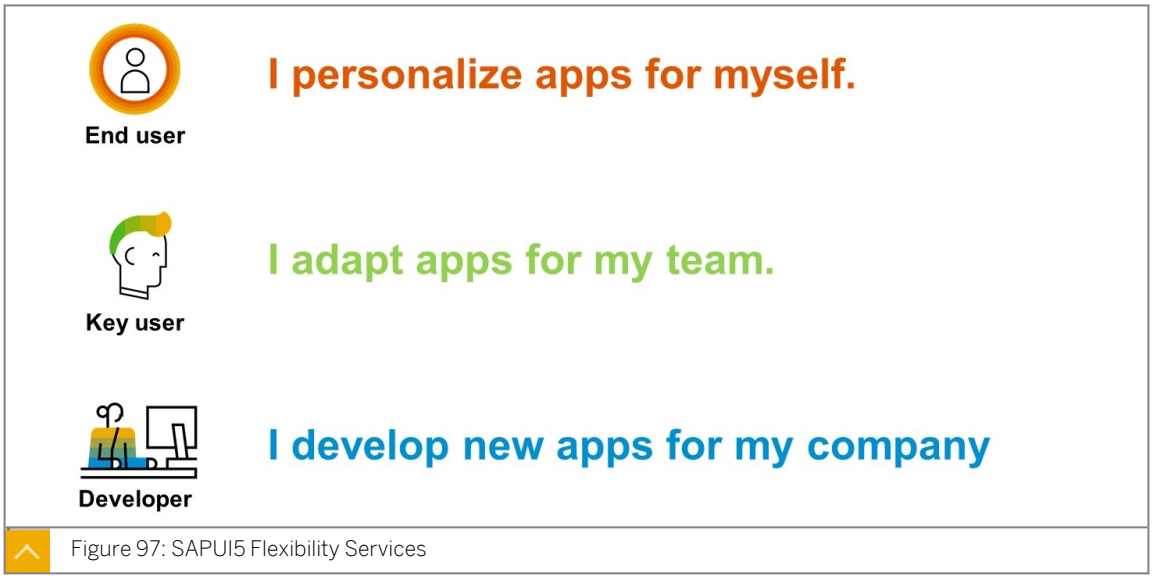


Figure 97: SAPUI5 Flexibility Services

SAPUI5 flexibility services enable functions for different user groups to personalize SAP Fiori apps, adapt their user interface at runtime, and develop new apps.

The service store user-specific data and client-wide data in a special repository on an ABAP server, which is called layered repository.

Prerequisites of SAPUI5 Flexibility Service

The following are prerequisites of the SAPUI5 flexibility service:

- Server: ABAP front end server and SAP NetWeaver 7.31 SP 11 or higher
- The following ICF node must be activated:
  - /SAP/BC/UI5_UI5
  - /SAP/BC/LREP
- For creating shared variants

© Copyright. All rights reserved.

- ABAP authorization object /UIF/FLEX with authorization field /UIF/KEYU = 'X' is required
- A transport request with a task assigned to the user is required

Currently, SAP Fiori supports variant management only when the back end system is SAP NetWeaver AS ABAP.

Variant Selection

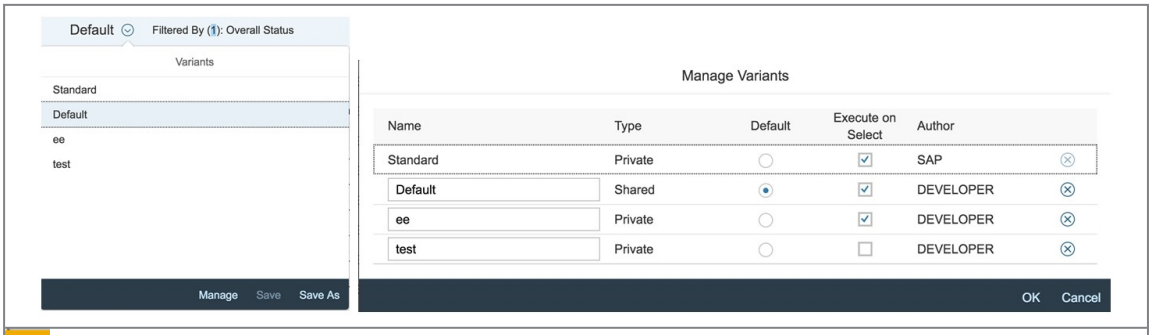


Figure 98: Selection of Variant

The figure, Selection of Variant, shows two screenshots of the two different ways to select variants.

The *Selection Variant* stores filter conditions for users.

After filtering the data, a user can save the condition as a variant. The following items are options for a selection variant:

- Name: Name of the variant
- Default: Select this variant when user entering the report
- Execute on Select: Execute query immediately after user select the variant
- Shared: Shared variant, a transport request must assigned when set a variant to shared

A user can manage their variants by clicking the *Manage* button.

© Copyright. All rights reserved.

Presentation Variant

- Table Variant store information about column list and order of the table
- After filter the data, user can save the condition as a variant, here are options for a table variant
  - Name: Name of the variant
  - Default: Select this variant when user entering the report
  - Shared: Shared variant, a transport request must assigned when set a variant to shared
- User can manage its variants by click Manage Button

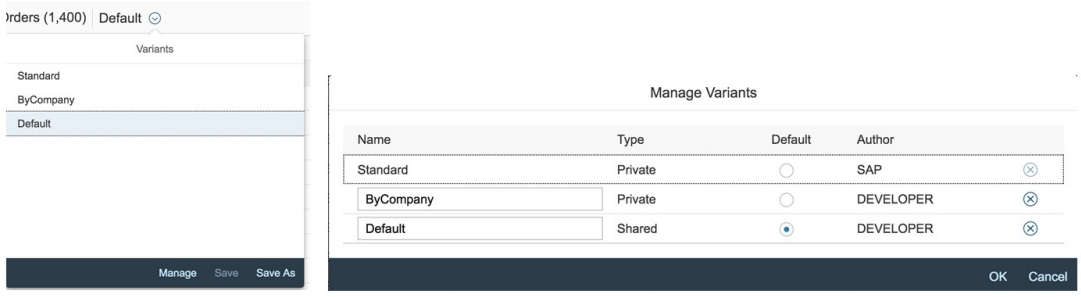


Figure 99: Presentation Variant

The figure, Presentation Variant, shows two screenshots of two different ways variants are presented.

Smart Variant Management

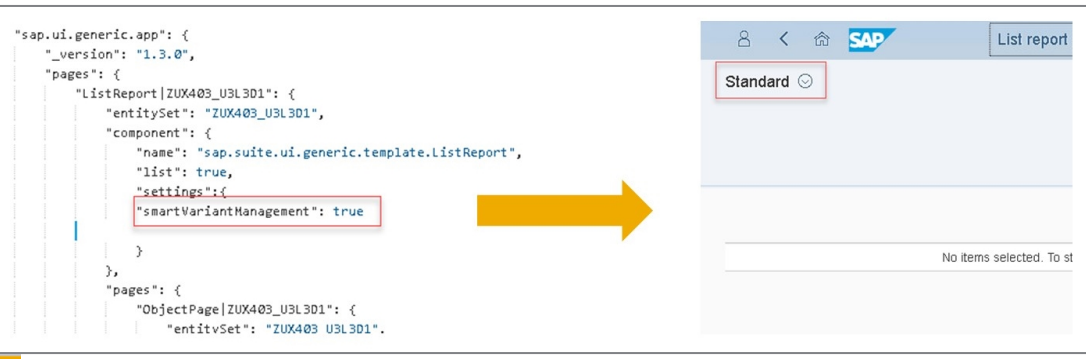


Figure 100: Smart Variant Management

To combine a selection variant and a presentation variant into one, smart variant management can be used.

To enable smart variant management, edit `manifest.json` in WebIDE, locate domain `sap.ui.generic.app`, modify the code as shown in the figure, Smart Variant Management.

© Copyright. All rights reserved.

Disable Variant Management

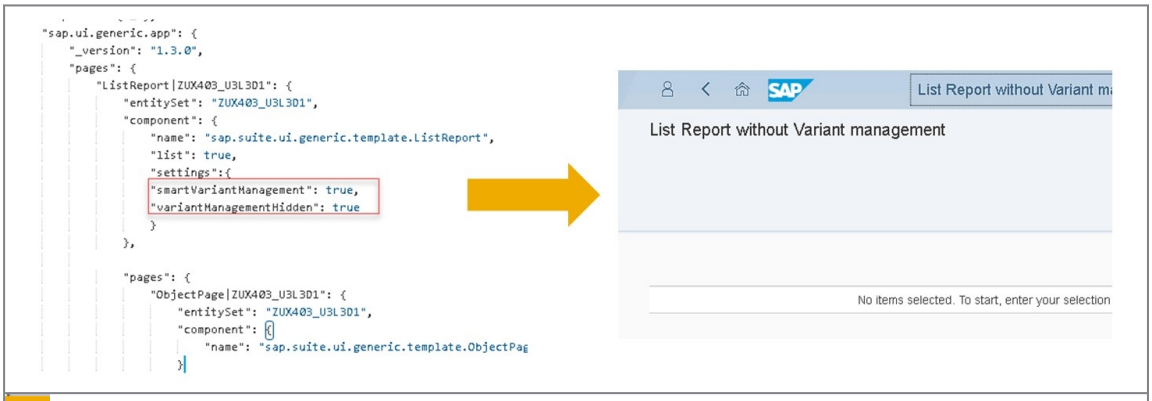


Figure 101: Disable Variant Management

To disable the variant management function, edit `manifest.json` in BAS, locate domain `sap.ui.generic.app`, and modify code as shown the figure, Disable Variant Management.

Transport Request for Shared Variants

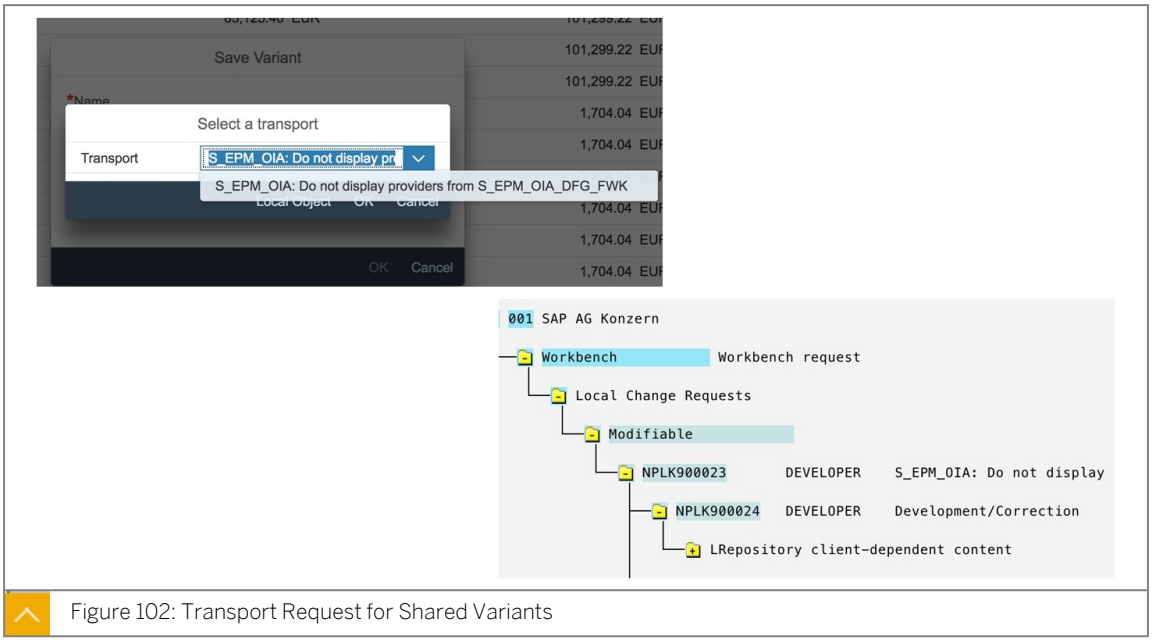


Figure 102: Transport Request for Shared Variants

A transport request, with a task assigned to the user, should be prepared before creating a shared variant. In `SE10`, you can find *LRepository* content in your task.

LRepository objects are client dependent, so either a workbench request or a customizing request is required.

LESSON SUMMARY

You should now be able to:

- Explain Variant Management

© Copyright. All rights reserved.

Learning Assessment

1. For a list report, which annotation is used to describe the name of business entity displayed on the report?

   Choose the correct answer.

   - ☐ A *@UI.lineItem.title*
   - ☐ B *@UI.headerInfo.type*
   - ☐ C *@UI.headerInfo.typeName*
   - ☐ D *@UI.headerInfo.typeNamePlural*

2. How can you hide some columns when the list report is accessed by a mobile phone?

   Choose the correct answer.

   - ☐ A Put all fields, which are not important at the end of the report. When the screen gets smaller, the fields will hide automatically.
   - ☐ B Add *@UI.hidden* for fields; not import.
   - ☐ C Prepare a different version of *@UI.lineItem* and assign them as a different qualifier.
   - ☐ D Set *UI.lineItem.importance* for those fields as **`#LOW`** or **`@Medium`**.

3. What is the use of the annotation @UI.hidden?

   Choose the correct answer.

   - ☐ A Prevent a column from display on the UI.
   - ☐ B Not expose these fields as a property of OData service.
   - ☐ C Prevent a column selected by user when customizing table settings.
   - ☐ D Create a invisible column to save the value in a hidden control of HTML.

© Copyright. All rights reserved.

4. Which of following description about search field is NOT true?

   Choose the correct answer.

   - ☐ A The search field is searching on more than one data field.
   - ☐ B There is only one search field per list report.
   - ☐ C The search field searches for only one data field.
   - ☐ D Search field support fuzzy search.

5. Choose available options for creating a value help for a selection field.

   Choose the correct answers.

   - ☐ A By adding a foreign key annotation.
   - ☐ B By adding a value help annotation.
   - ☐ C If the domain which associates to the field, has a fixed value, the value help will be generated automatically.
   - ☐ D By adding annotations to list all possible in source code of CDS.

6. When running a list report, users can only filter data using fields provided as selection fields by developer.

   Determine whether this statement is true or false.

   - ☐ True
   - ☐ False

7. What steps needs to be done in *manifest.json* if you need to hide variant function in a list report?

   Choose the correct answers.

   - ☐ A Add setting to enable smart Variant Management.
   - ☐ B Add setting to disable smart Variant Management.
   - ☐ C Set setting *variantManagementHidden* to true.
   - ☐ D Set setting *showVariantManagement* to false.

© Copyright. All rights reserved.

Learning Assessment - Answers

1. For a list report, which annotation is used to describe the name of business entity displayed on the report?

   Choose the correct answer.

   - [ ] A *@UI.lineItem.title*
   - [ ] B *@UI.headerInfo.type*
   - [ ] C *@UI.headerInfo.typeName*
   - [x] D *@UI.headerInfo.typeNamePlural*

   Correct. For a list report, *@UI.headerInfo.typeNamePlural* is used to describe the name of business entity displayed on the report.

2. How can you hide some columns when the list report is accessed by a mobile phone?

   Choose the correct answer.

   - [ ] A Put all fields, which are not important at the end of the report. When the screen gets smaller, the fields will hide automatically.
   - [ ] B Add *@UI.hidden* for fields; not import.
   - [ ] C Prepare a different version of *@UI.lineItem* and assign them as a different qualifier.
   - [x] D Set *UI.lineItem.importance* for those fields as **`#LOW`** or **`@Medium`**.

   Correct. To hide some columns, set *UI.lineItem.importance* for those fields as **`#LOW`** or **`@Medium`**.

3. What is the use of the annotation @UI.hidden?

   Choose the correct answer.

   - [ ] A Prevent a column from display on the UI.
   - [ ] B Not expose these fields as a property of OData service.
   - [x] C Prevent a column selected by user when customizing table settings.
   - [ ] D Create a invisible column to save the value in a hidden control of HTML.

   Correct. @UI.hidden is used to prevent a column selected by user when customizing table settings.

© Copyright. All rights reserved.

4. Which of following description about search field is NOT true?

   Choose the correct answer.

   - [ ] A The search field is searching on more than one data field.
   - [ ] B There is only one search field per list report.
   - [x] C The search field searches for only one data field.
   - [ ] D Search field support fuzzy search.

   Correct. The search field searches for only one data field, is not true.

5. Choose available options for creating a value help for a selection field.

   Choose the correct answers.

   - [x] A By adding a foreign key annotation.
   - [x] B By adding a value help annotation.
   - [ ] C If the domain which associates to the field, has a fixed value, the value help will be generated automatically.
   - [ ] D By adding annotations to list all possible in source code of CDS.

   Correct. A value help for a selection field can be created by adding a foreign key annotation and by adding a value help annotation.

6. When running a list report, users can only filter data using fields provided as selection fields by developer.

   Determine whether this statement is true or false.

   - [ ] True
   - [x] False

   Correct. When running a list report, users can filter data using other fields than provided as selection fields by developer.

© Copyright. All rights reserved.

7. What steps needs to be done in *manifest.json* if you need to hide variant function in a list report?

   Choose the correct answers.

   [X] A Add setting to enable smart Variant Management.

   [ ] B Add setting to disable smart Variant Management.

   [X] C Set setting *variantManagementHidden* to true.

   [ ] D Set setting *showVariantManagement* to false.

   Correct. To hide variant function in a list report, add setting to enable smart Variant Management and set setting *variantManagementHidden* to true.

© Copyright. All rights reserved.

© Copyright. All rights reserved.

UNIT 4 Object Page

UNIT OBJECTIVES

- Use Basic Annotations for Object Pages
- Use Header Facets for Object Pages
- Use Sections and Facets in Object Pages

© Copyright. All rights reserved.

© Copyright. All rights reserved.

Unit 4
Lesson 1

Using Basic Annotations for Object Pages

LESSON OBJECTIVES

After completing this lesson, you will be able to:

- Use Basic Annotations for Object Pages

Basic Annotations for Object Pages

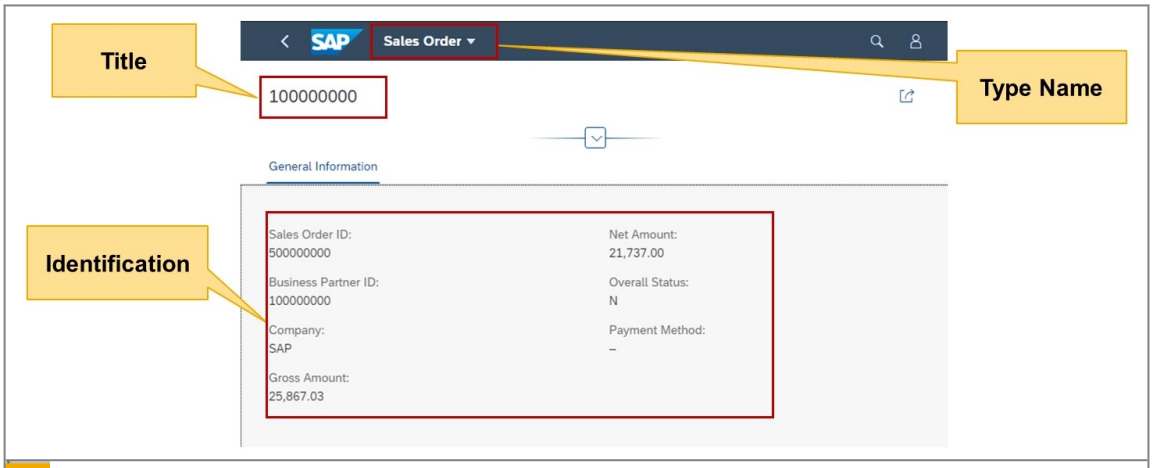


Figure 103: A Simple Object Page

An object page displays when an end user selects a row in a list report.

A basic object page contains a header area and a body.

In the top, a name describes the type of objects displays in the center.

The header area contains title, description, and image of the record. The value should bind to a property of the OData service.

In the body area, by default, fields annotated with `@UI.identification` display.

Annotations for Basic Object Page

Essential Annotations for basic object pages are:

@UI.headerinfo.typeName
Indicates the type name of the business object.

@UI.headerinfo.title.value
Points to the ID data field.

@UI.headerinfo.description.value
Point to the Description data field.

@UI.identification

© Copyright. All rights reserved.

Like `UI.LineItem`, fields with this annotation displays in the *General Information* section of the page.

Example of a Business Object Page

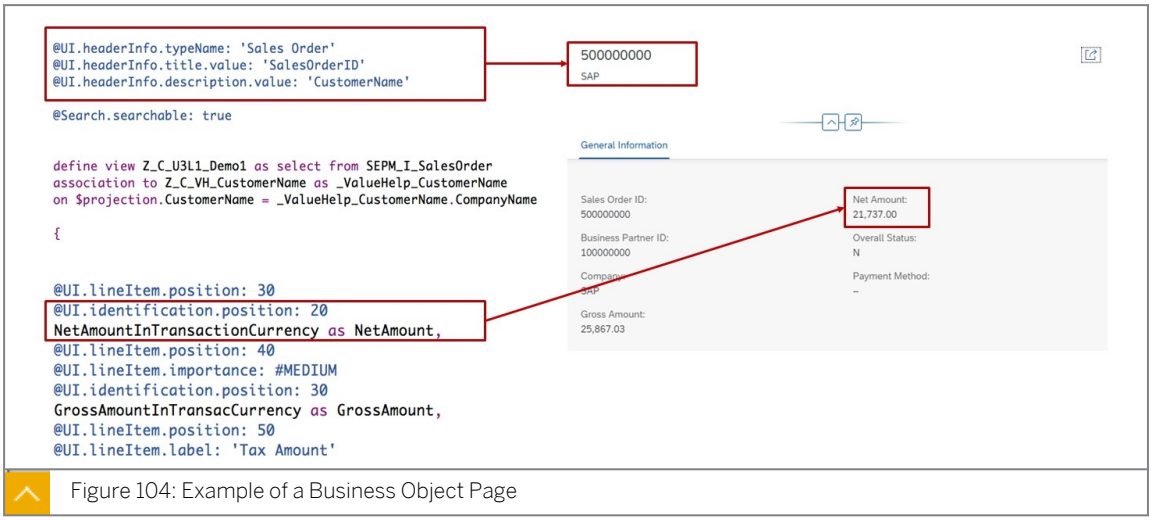


Figure 104: Example of a Business Object Page

The figure, Example of a Business Object Page, displays an example for a simple object page. From this figure, you can see how `@UI.headerInfo` and `@UI.identification` display on the Object Page.

Please notice that the `@UI.identification` is only shown when a facet is implemented (see Lesson 3 of this unit).

LESSON SUMMARY

You should now be able to:

- Use Basic Annotations for Object Pages

© Copyright. All rights reserved.

Using Header Facets for Object Pages

LESSON OBJECTIVES
After completing this lesson, you will be able to:

- Use Header Facets for Object Pages

Header Facets for Object Pages

Define Header Facets

Header Facets

Pieces of information placed on header part of object page

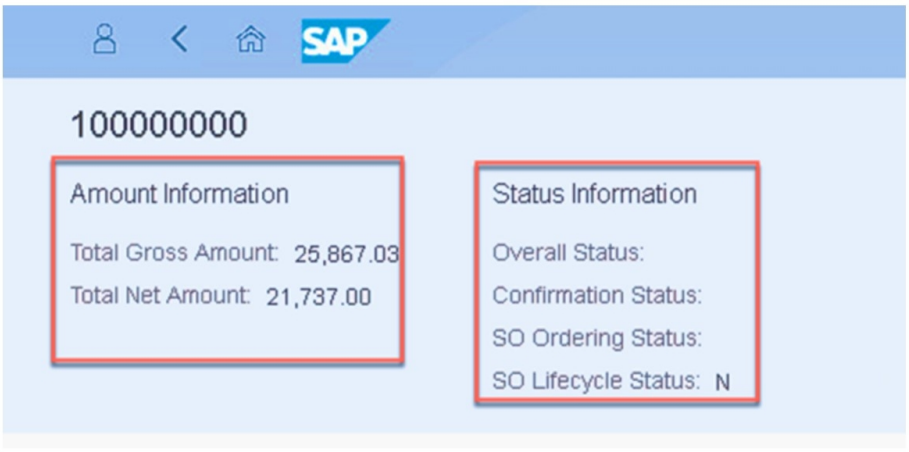


Figure 105: Define Header Facets

Header facets are pieces of information placed on the header part of an Object Page.

Generally, key information and status information should be put in this area.

© Copyright. All rights reserved.

Types of Header Facets

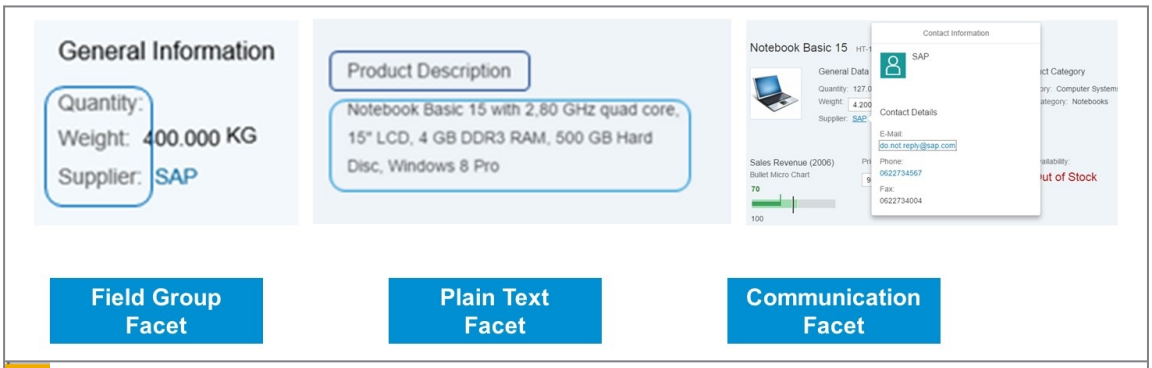


Figure 106: Types of Header Facets (1)

Field Group Facet displays a group of fields with a label.

Plain Text Facet displays description information from exactly one field.

Communication Facet generates a hyper link. The user can access communication information (for example, phone, email, and so on) by selecting the link.

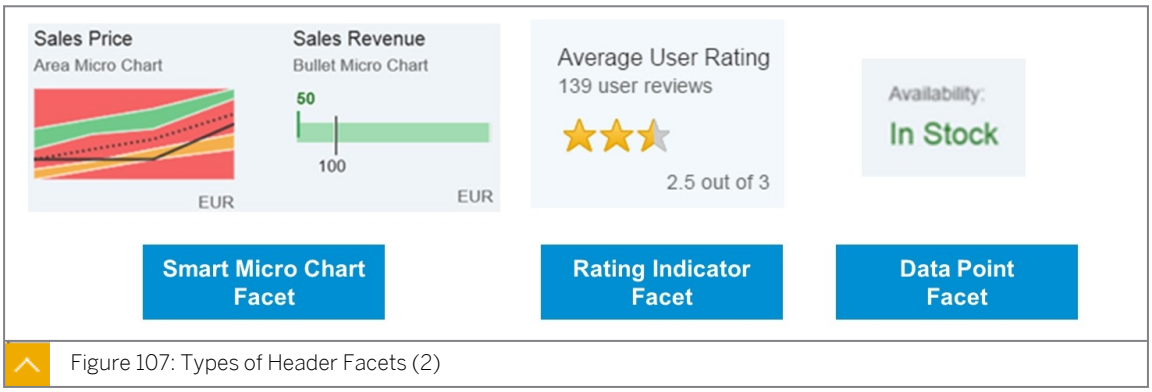


Figure 107: Types of Header Facets (2)

Smart Micro Chart Facet displays a micro chart to help the user get the most important KPI information.

Rating Indicator Facet displays rating information as stars.

Data Point Facet display a key figure or status information in different color or style to help user get the meaning at a glance.

Note:
Smart Micro Chart Facet, Rating Indicator Facet, and Data Point Facet will be covered in later chapter of this course.

For a full list of types of header facets, open SAPUI5 SDK and navigate to following path.

Documentation->Developing Apps with SAP Fiori Elements->List Report and Object Page->Configuring Object Page Features->Setting up the Object Page Header->Header Facets.

© Copyright. All rights reserved.

Field Group Facets

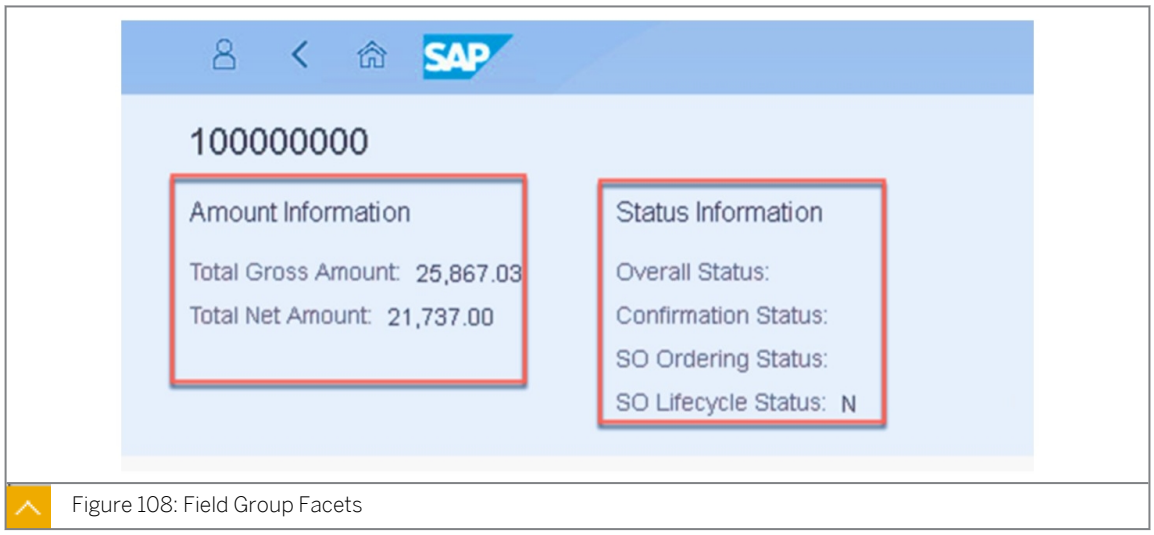


Figure 108: Field Group Facets

A field group defines the fields displayed within a specific reference facet. For example, the figure shows the following fields within the Amount Information and Status Information:

- Amount Information:
  - Total Gross Amount
  - Total Net Amount
  - Status Information
- Overall Status:
  - Confirmation Status
  - SO Ordering StatusSO Lifecycle Status

Define Field Groups

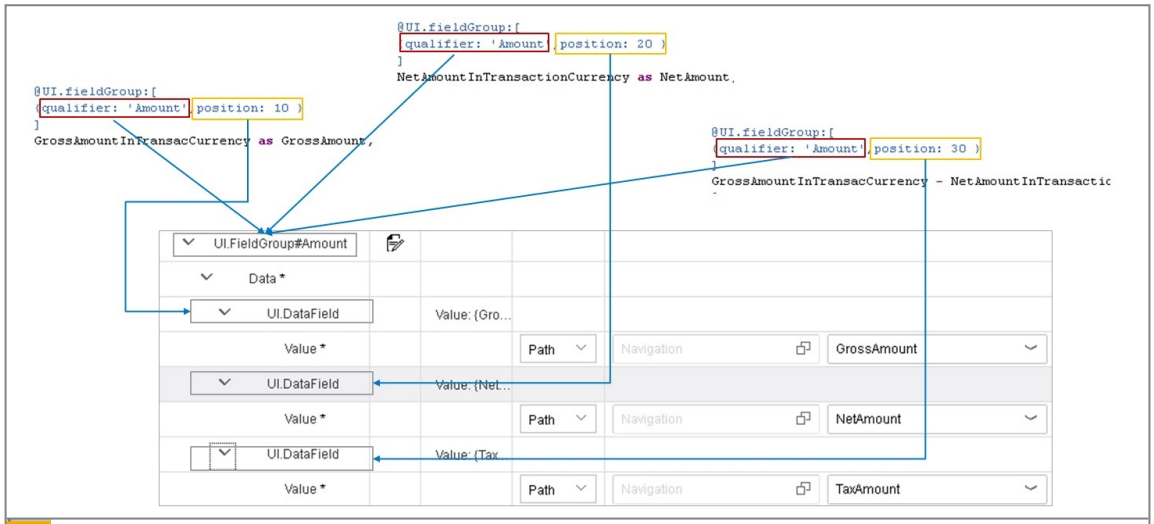


Figure 109: Define Field Groups

© Copyright. All rights reserved.

Term UI.FieldGroup is used to define field groups. Because, in most cases, a CDS view has more than one field groups, a qualifier should be assigned to each field group to distinguish among those field groups.

A field can belong to different field groups, so that the UI.fieldGroup annotation for each field is an array. The annotation should use JSON format for easier modification in the future.

When translating the field groups annotation to OData annotation, all @UI.fieldGroup annotations that have the same qualifier will be combined to one term. The sequence of fields in a field group will be determined by @UI.fieldGroup.position defined in CDS view.

Define Field Group Header Facets

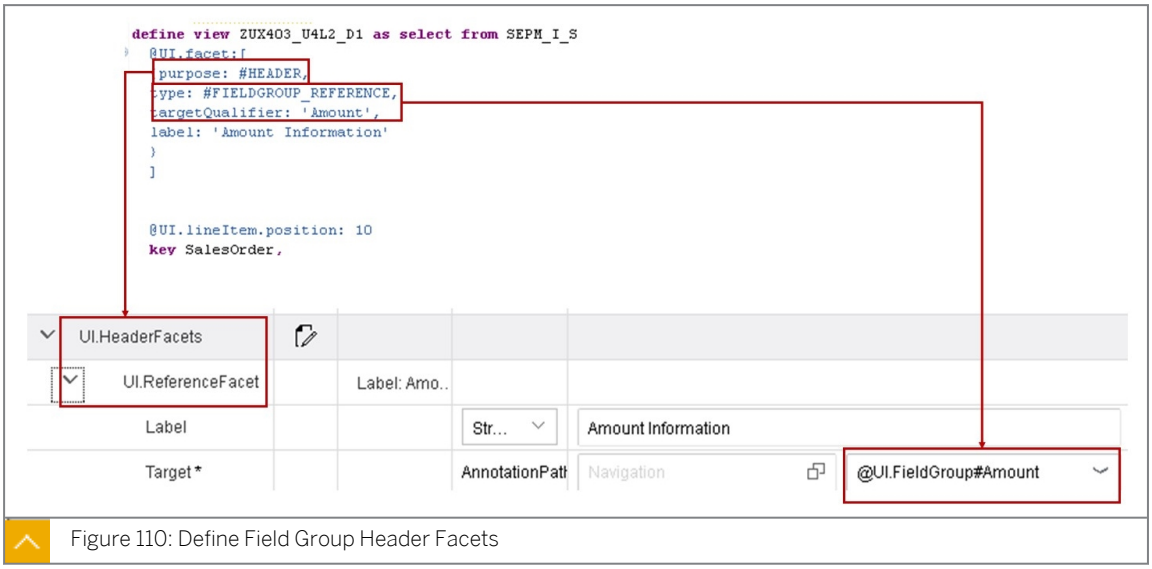


Figure 110: Define Field Group Header Facets

To define a field group header facet, a *@UI.facet* annotation should be added for the first field of your CDS view. The *@UI.facet* annotation is an array and should contains all facet definitions.

For Field Group Header Facet, it should do the following:

- Set purpose to #HEADER.
- Set type to #FIELDGROUP_REFERENCE.
- Set targetQualifier to the qualifier of corresponding field group.
- Set position and label.

The translated OData annotation will have a term called UI.HeaderFacet and have some UI.ReferenceFacet in it.

A more common way is to define *UI.fieldGroup* in CDS view, since field group is data intensive information, but define *UI.HeaderFacet* annotation as Local Annotation while facets are more relevant with UI layer. Another reason for this is *@UI.facet* annotation in CDS view is only supported by NetWeaver AS ABAP 7.52.

© Copyright. All rights reserved.

Defining a Plain Text Header Facet

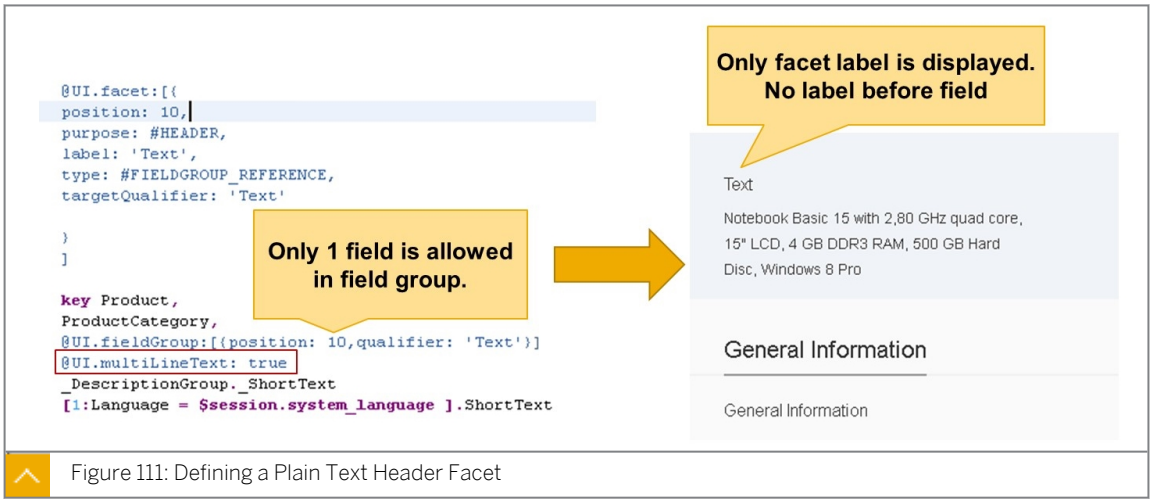


Figure 111: Defining a Plain Text Header Facet

The Plain Text Header Facet is quite like the field group header facet.

To define a plain text header facet, you should do the following:

- Define a field group containing only one field.
- Set the field as a multiline text.

Definition of the Header Facet Communication

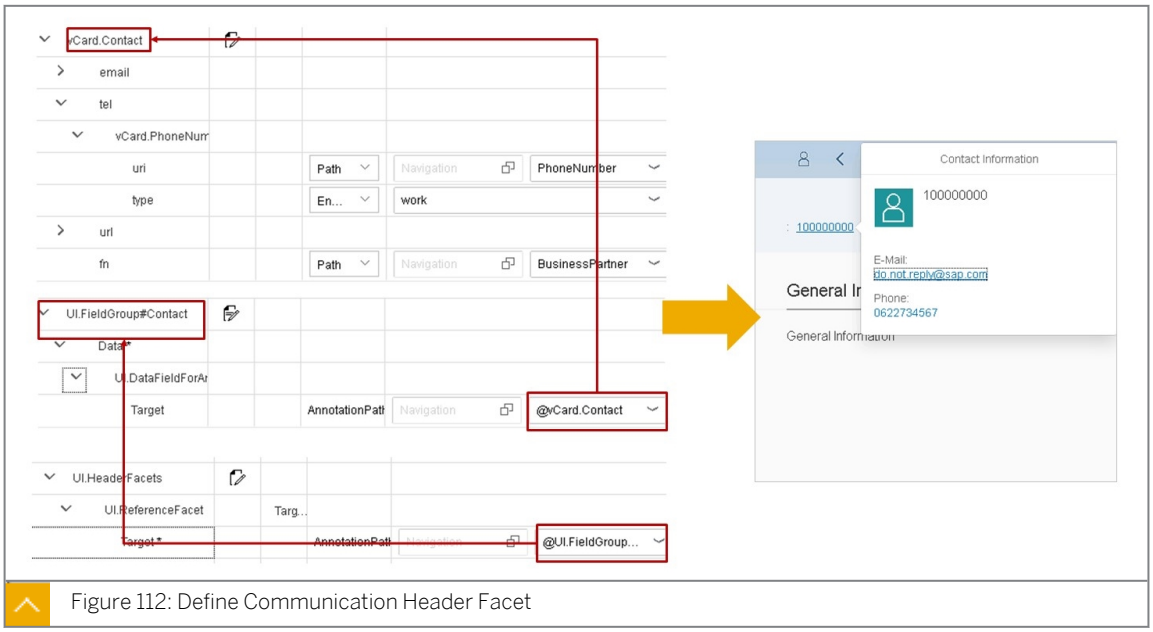


Figure 112: Define Communication Header Facet

Technically, a communication header facet is a Field Group header facet. The only difference is at least one field of the field group should reference to a *vCard.Contact* annotation.

Currently, a communication header facet can only be created by add local annotation through annotation modeler in SAP WebIDE.

The steps are:

© Copyright. All rights reserved.

1. Add a *vCard.Contact* term and bind properties to corresponding fields. Mention that for properties like phone or email, a type property is needed to clarify what is the usage of the phone number, or email address (for example, home, work, and so on).
2. Add a *UI.FieldGroup* term with a *DataFieldForAnnotation* item, reference to the *vCard.Contact* annotation.
3. Add a *UI.HeaderFacet* term, then add a *ReferenceFacet* in it. Reference the facet to the field group.

Arrangement of Header Facets

Figure 113: Arrangement of Header Facets

If there is more than one Header Facet, they will arrange in a row and will start a new row if there is not enough room. Here is an example of the arrangement for eight Header Facets.

LESSON SUMMARY

You should now be able to:

- Use Header Facets for Object Pages

© Copyright. All rights reserved.

Unit 4
Lesson 3

Using Sections and Facets in Object Pages

LESSON OBJECTIVES

After completing this lesson, you will be able to:

- Use Sections and Facets in Object Pages

Sections and Facets in Object Pages

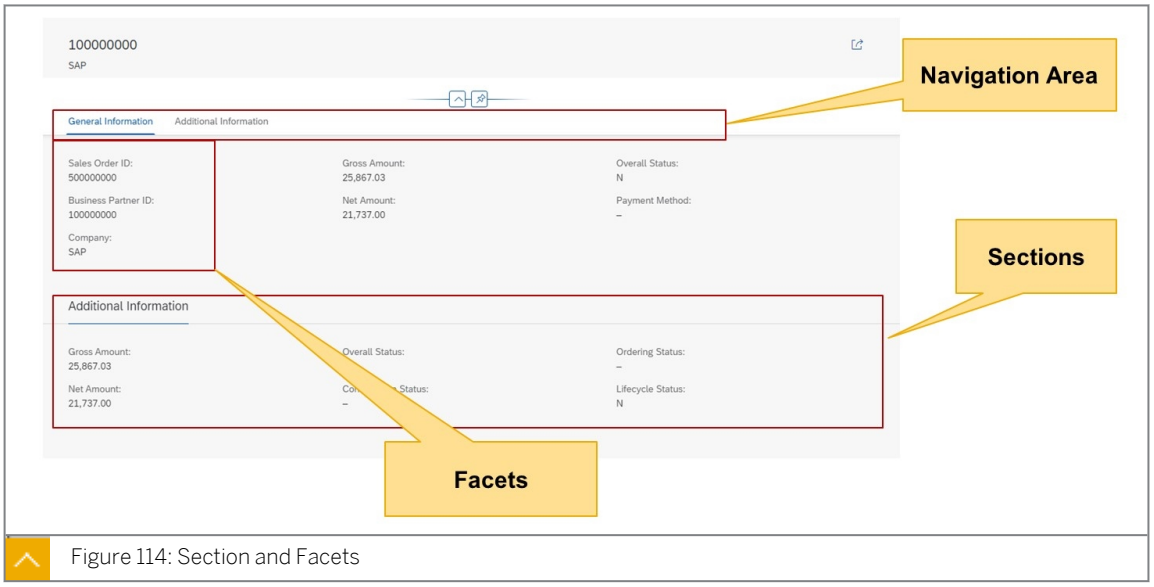


Figure 114: Section and Facets

The areas of the Object Page below the header facets are facets grouped by sections. There is also a navigation bar for navigation to specific sections.

© Copyright. All rights reserved.

Types of Facets in Sections

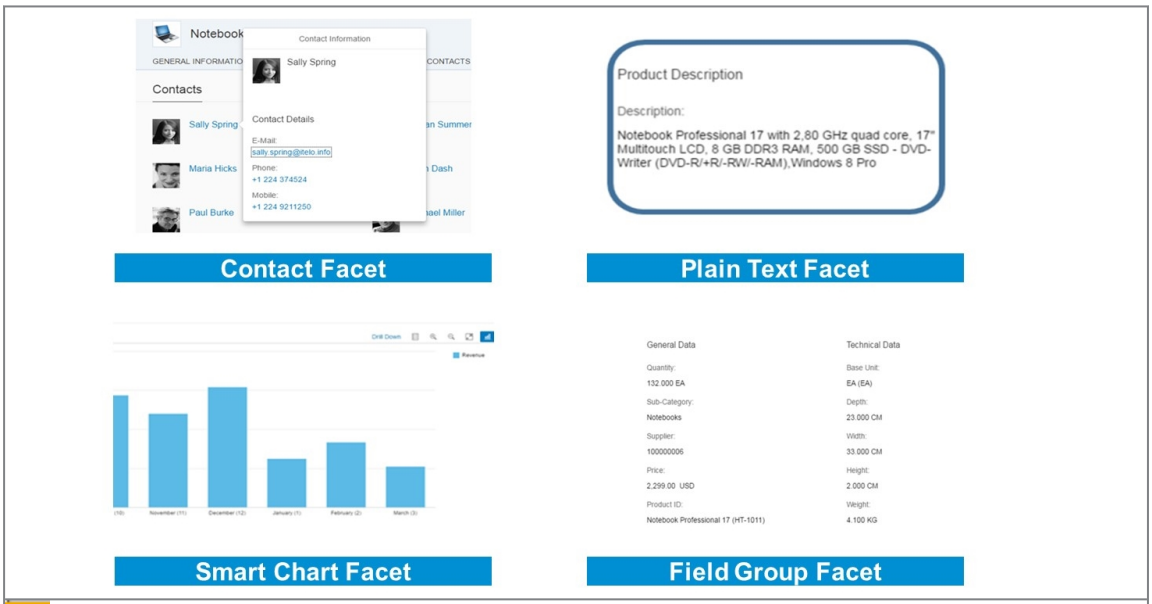


Figure 115: Types of Facets in Sections

The following items outline the details of the various types of facets:

Contact Facet

Displays contacts information from a 1 to * association.

Plain Text Facet

Displays a long description.

Smart Chart Facet

Displays a chart. The data usually come from a 1 to * association.

Field Group Facet

Display field groups in a section.

Note:
A detailed explanation for smart chart facets is given in a later lesson of this course.

Detailed information about sections, indicator facets, and data point facets is given in a later chapter of this course.

For a full list of the types of header facets, open the SAPUI5 SDK and navigate to the following path: *Documentation → Developing Apps with SAP Fiori Elements → List Report and Object Page → Configuring Object Page Features → Defining and Adapting Sections*.

© Copyright. All rights reserved.

Field Group Facets-CDS Annotation

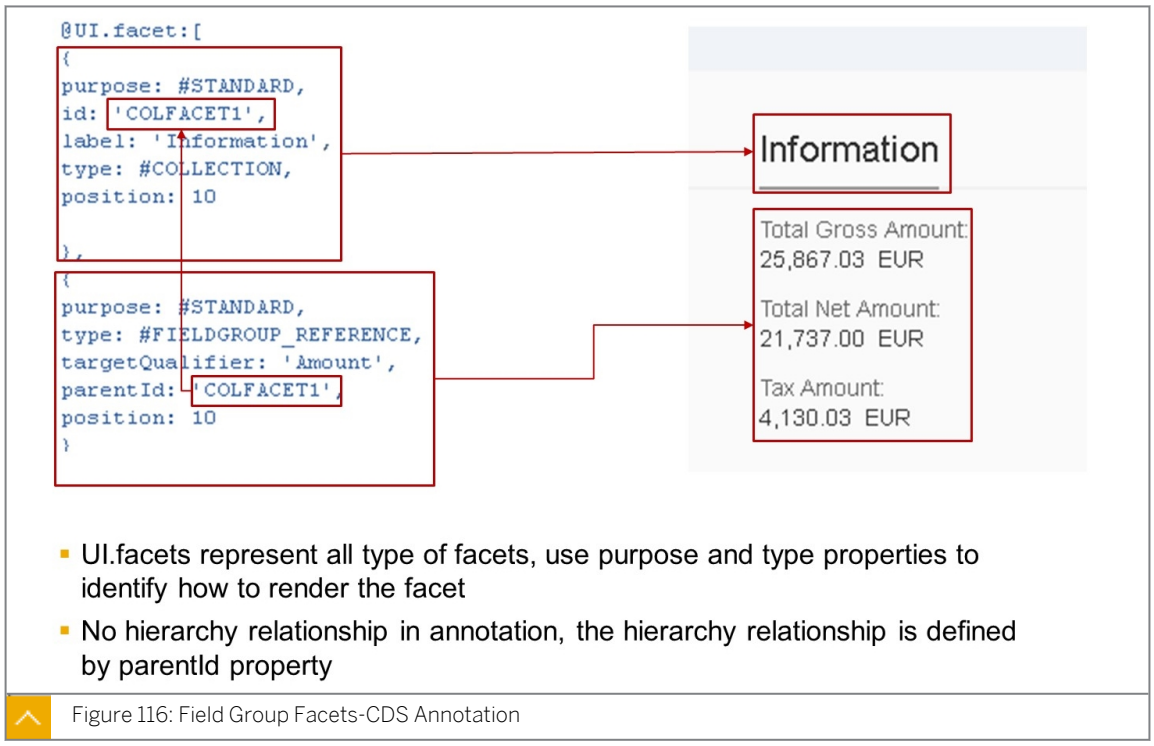


Figure 116: Field Group Facets-CDS Annotation

A field group facet must be put into a section.

To create a section, use the *UI.facet* annotation to create a facet with type *#COLLECTION* and assign it an ID.

To create a field group facet, add a second record in the*UI.facet* with a type *@FIELDGROUP_REFERENCE* and a set *targetQualifier* to **`qualifier of the field group`**. Then add a reference to the reference of the collection facet by *parentId*.

> Note:
> Defining facets in CDS view is only supported by ABAP 7.52.

Field Group Facets-Local Annotation

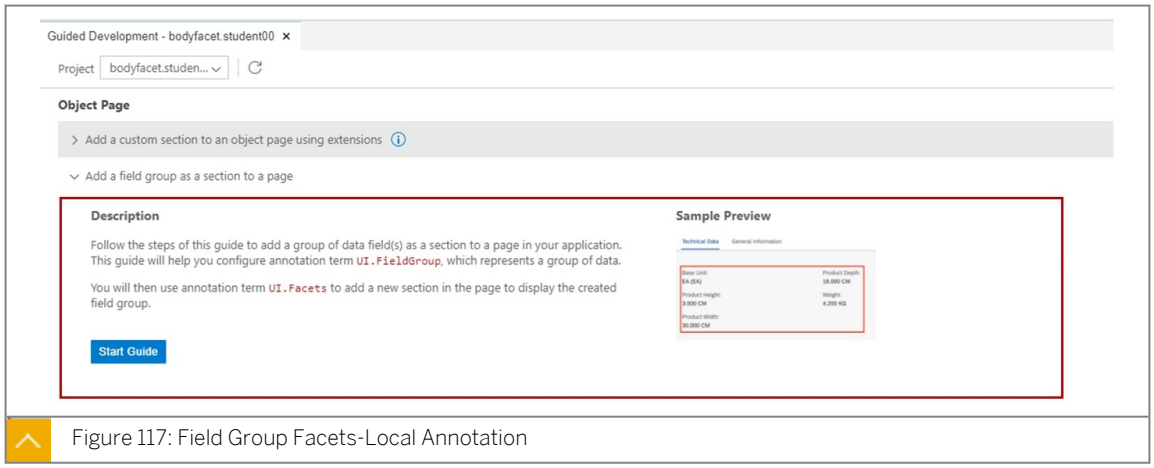


Figure 117: Field Group Facets-Local Annotation

© Copyright. All rights reserved.

Back end systems do not support the definition of facets on the back end. Local annotation can be used to define facets. Guided Development will help you creating a facet and add the facet (if needed) to a newly created section.

There are different terms for the collection facet and the reference facet. The steps are:

1. Create a UI.CollectionFacet.
2. Create a UI.ReferenceFacet under the collection facet and reference it to field group annotation.

Arrangement of Sections and Facets

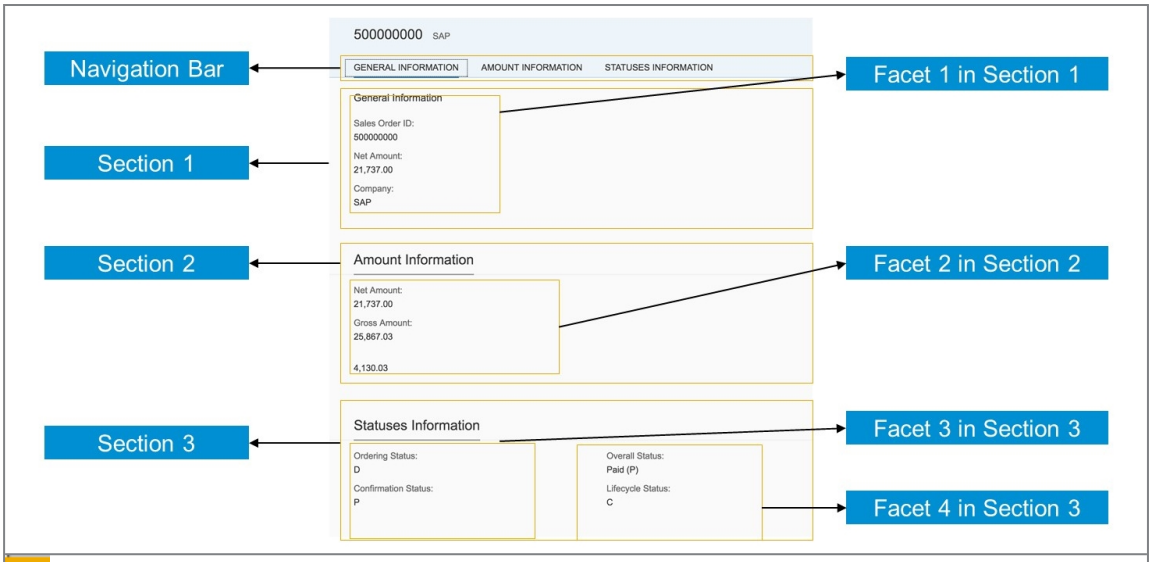
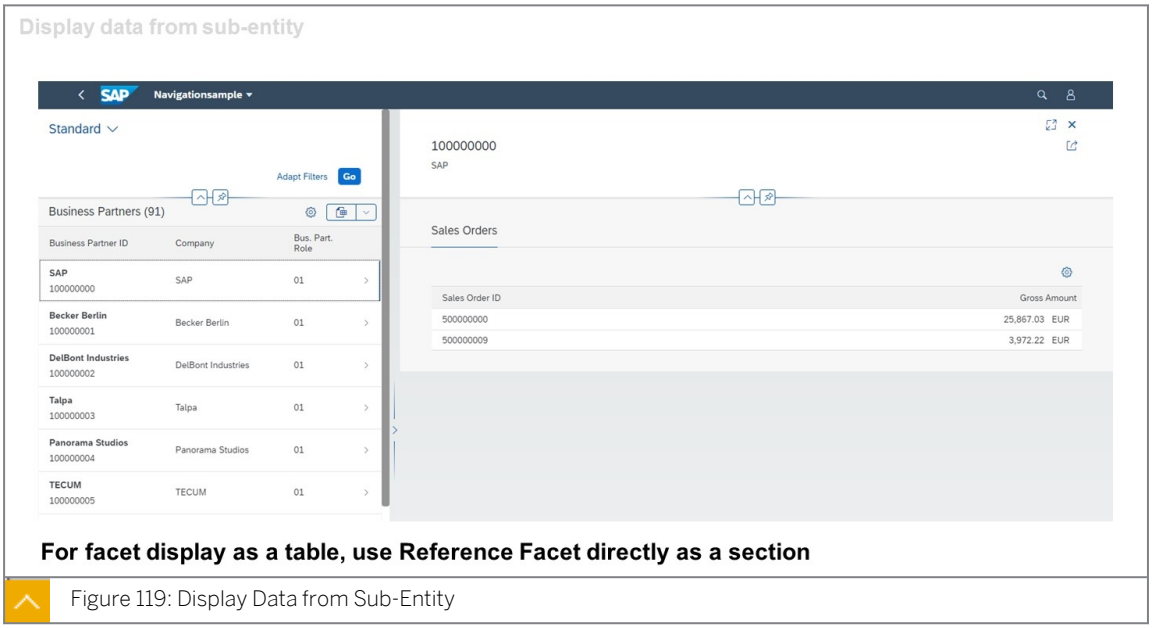


Figure 118: Arrangement of Sections and Facets

All sections arrange vertically.

All facets in a section arrange in a row.

Display Data from Sub-Entity



For facet display as a table, use Reference Facet directly as a section

Figure 119: Display Data from Sub-Entity

© Copyright. All rights reserved.

If you need to display data from a 1 to * association as a table in an object page, you should use a reference facet to create a section, since the section can only contain one facet.

Steps for Facets to Display the Data of a Sub-Entity

Figure 120: Steps for Facets to Display the Data of a Sub-Entity

The following steps allow facets to display the data of a sub-entity:

1. Create a CDS view for detailed data. (for example, Sales order items data).
2. In the CDS view for the list report, create a to * association to the CDS view created in step 1.
3. Expose the association by writing it's name like a field.
4. Create a facet with type *#LINEITEM_REFERENCE* and link it to the association by setting the *targetElement* property.

Display of Sub-Entity's Data - Local Annotation

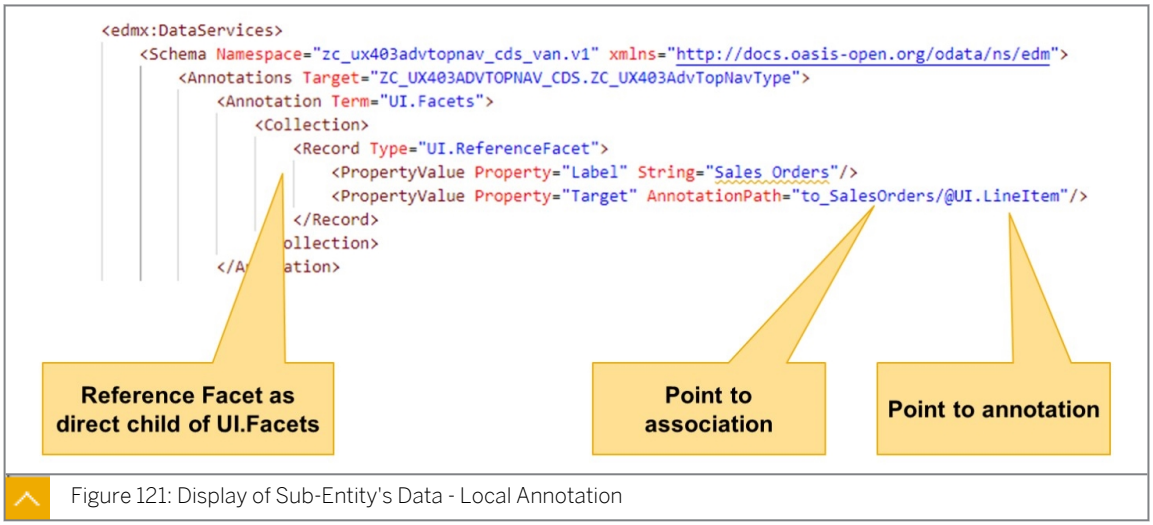


Figure 121: Display of Sub-Entity's Data - Local Annotation

For local annotation, you need to create *UI.ReferenceFact* as a direct child of *UI.Facets (No collection facet)*.

For the reference facet, you need to set the target by referencing the association and annotation.

© Copyright. All rights reserved.

Note that when exposing a CDS view as an OData service, SAP added "to" before CDS association name as OData association name. For adding new sections developers can use the guided development feature of SAP Business Application Studio.

LESSON SUMMARY

You should now be able to:

- Use Sections and Facets in Object Pages

© Copyright. All rights reserved.

Learning Assessment

1. How do you set a field as the title for a business entity?

   Choose the correct answer.

   - ☐ **A** Add *@UI.title* annotation for that field.
   - ☐ **B** Add *@UI.headerInfo.title* annotation for that field.
   - ☐ **C** Add *@UI.headerInfo.title.value* for the CDS view and reference it to the field.
   - ☐ **D** Add *@UI.headerInfo.title* for the CDS view and reference it to the field.

2. Which types are supported as header facets for an object page?

   Choose the correct answers.

   - ☐ **A** Field Group Facet
   - ☐ **B** Plan Text Facet
   - ☐ **C** Smart Chart Facet
   - ☐ **D** Rating Indicator Facet

3. To add a header facet, you should use `@UI.HeaderFacet` annotation and put it before the define statement of your CDS view.

   Determine whether this statement is true or false.

   - ☐ True
   - ☐ False

4. The CDS annotation `@UI.facet` is used to:

   Choose the correct answers.

   - ☐ **A** Create a Header Facet.
   - ☐ **B** Create a collection facet, which is displayed as section.
   - ☐ **C** Create a reference under collection facet.
   - ☐ **D** Create content, like field groups, or charts for a Facet.

© Copyright. All rights reserved.

5. Which of the following properties are relevant to the type of facet?

   Choose the correct answers.

   - ☐ A purpose
   - ☐ B targetQualifier
   - ☐ C parentId
   - ☐ D type

6. If your back end system is based on ABAP 7.50, which are available ways of using facets?

   Choose the correct answers.

   - ☐ A Declare both: field groups and facets in CDS view.
   - ☐ B Declare both: field group and facet as local annotation using BAS.
   - ☐ C Declare field group in CDS view, declare facets as local annotation using BAS.
   - ☐ D It is not possible to use facets in ABAP 7.50.

© Copyright. All rights reserved.

Learning Assessment - Answers

1. How do you set a field as the title for a business entity?

   Choose the correct answer.

   - [ ] A Add @*UI.title* annotation for that field.
   - [ ] B Add @*UI.headerInfo.title* annotation for that field.
   - [x] C Add @*UI.headerInfo.title.value* for the CDS view and reference it to the field.
   - [ ] D Add @*UI.headerInfo.title* for the CDS view and reference it to the field.

   Correct. To set a field as the title for a business entity, add @*UI.headerInfo.title.value* for the CDS view and reference it to the field.

2. Which types are supported as header facets for an object page?

   Choose the correct answers.

   - [x] A Field Group Facet
   - [x] B Plan Text Facet
   - [ ] C Smart Chart Facet
   - [x] D Rating Indicator Facet

   Correct. Field Group Facet, Plan Text Facet, and Rating Indicator Facet are supported as header facets for an object page.

3. To add a header facet, you should use `@UI.HeaderFacet` annotation and put it before the define statement of your CDS view.

   Determine whether this statement is true or false.

   - [ ] True
   - [x] False

   Correct. To add a header facet, you should not use `@UI.HeaderFacet` annotation and put it before the define statement of your CDS view.

© Copyright. All rights reserved.

4. The CDS annotation `@UI.facet` is used to:

   Choose the correct answers.

   - [X] A Create a Header Facet.
   - [X] B Create a collection facet, which is displayed as section.
   - [X] C Create a reference under collection facet.
   - [ ] D Create content, like field groups, or charts for a Facet.

   Correct. The CDS annotation `@UI.facet` is used to, Create a Header Facet, create a collection facet, which is displayed as section, and create a reference under collection facet.

5. Which of the following properties are relevant to the type of facet?

   Choose the correct answers.

   - [X] A purpose
   - [ ] B targetQualifier
   - [ ] C parentId
   - [X] D type

   Correct. Purpose and type are the relevant properties to the type of facet.

6. If your back end system is based on ABAP 7.50, which are available ways of using facets?

   Choose the correct answers.

   - [ ] A Declare both: field groups and facets in CDS view.
   - [X] B Declare both: field group and facet as local annotation using BAS.
   - [X] C Declare field group in CDS view, declare facets as local annotation using BAS.
   - [ ] D It is not possible to use facets in ABAP 7.50.

   Correct. To use facets, declare both: field group and facet as local annotation using BAS. And declare field group in CDS view, declare facets as local annotation using BAS.

© Copyright. All rights reserved.

UNIT 5 Advanced Topics of List Report and Object Page

UNIT OBJECTIVES

- Explain the navigation concept and annotations
- Describe options of external navigation
- Use data visualization
- Create Charts
- Perform CRUD operations with BOPF

© Copyright. All rights reserved.

© Copyright. All rights reserved.

Unit 5
Lesson 1

Explaining Navigation Concept and Annotations

LESSON OBJECTIVES

After completing this lesson, you will be able to:

- Explain the navigation concept and annotations
- Describe options of external navigation

Navigation Concept and Annotations

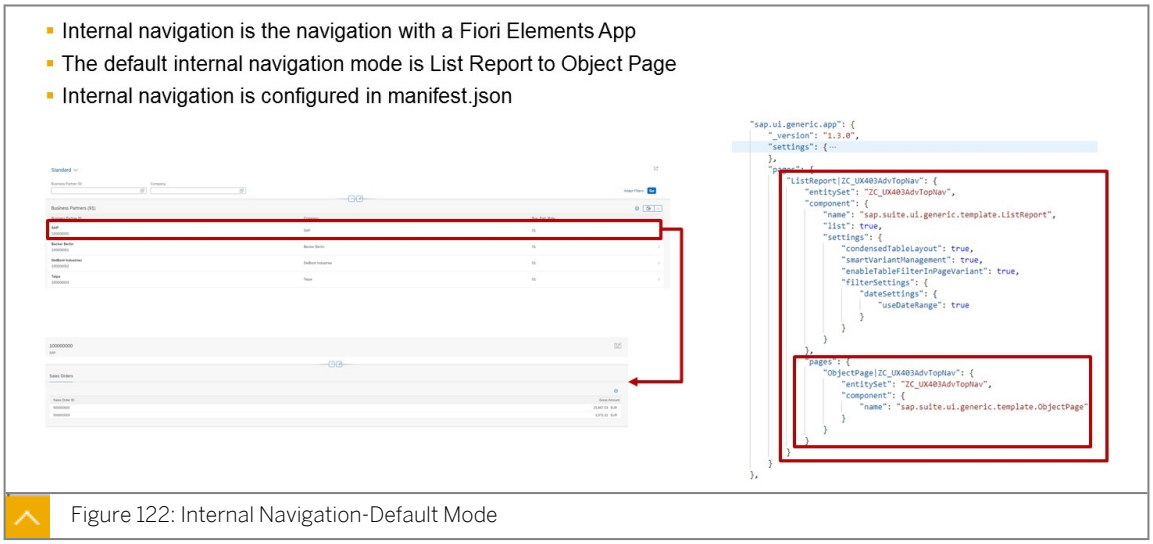


Figure 122: Internal Navigation-Default Mode

In the *manifest.json* file, you can define which pages are available in the app. At the top level, array "pages" are defined. This array should have only one point of entry, which is the main entry point to the app and should always be a list report. Each page can have array "pages" containing all sub-pages of the given page. Only one sub-page is allowed in the list report. This should be an object page for the same entity set. By default, the object page does not contain any sub pages.

© Copyright. All rights reserved.

Internal Navigation-List Report Only

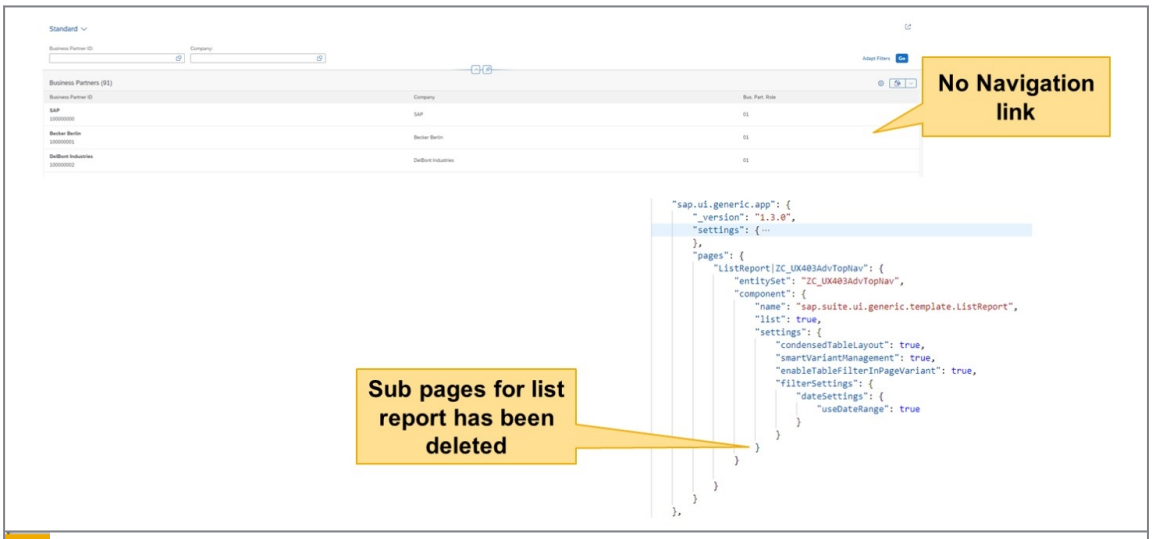


Figure 123: Internal Navigation-List Report Only

If you want to display a list report without an object page, you can delete **pages** and all properties in it under *ListReport* → *<Your EntitySetName>*.

Subpages for Object Page

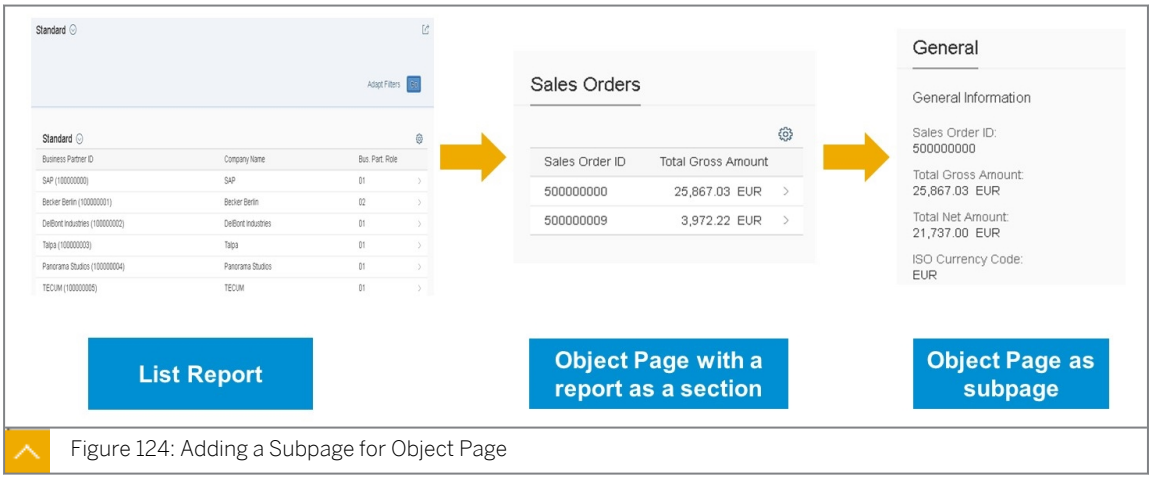


Figure 124: Adding a Subpage for Object Page

You can add another page as a subpage to your object page. This lets you navigate to a very detailed level of information.

© Copyright. All rights reserved.

Adding a Subpage for Object Page-manifest.json

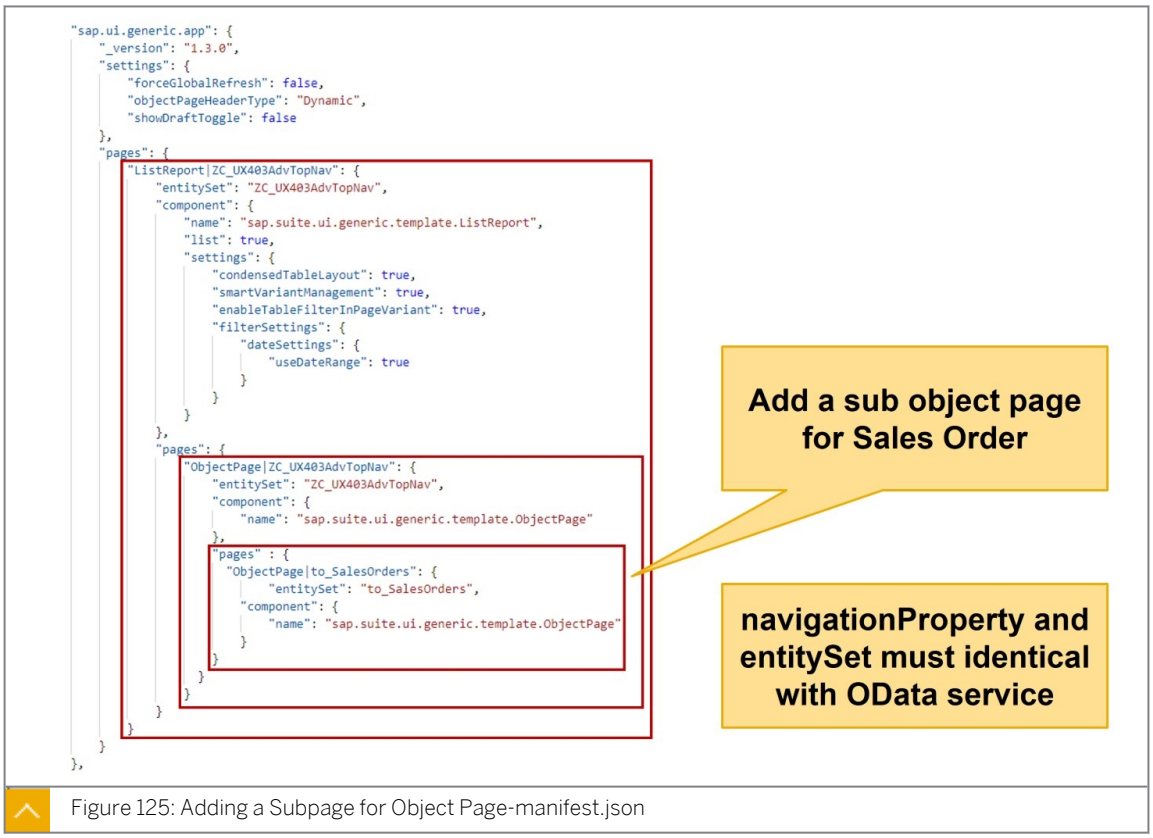


Figure 125: Adding a Subpage for Object Page-manifest.json

Under the *ObjectPage* node, you can add a *pages* sub node and add a subpage for each OData entity set.

© Copyright. All rights reserved.

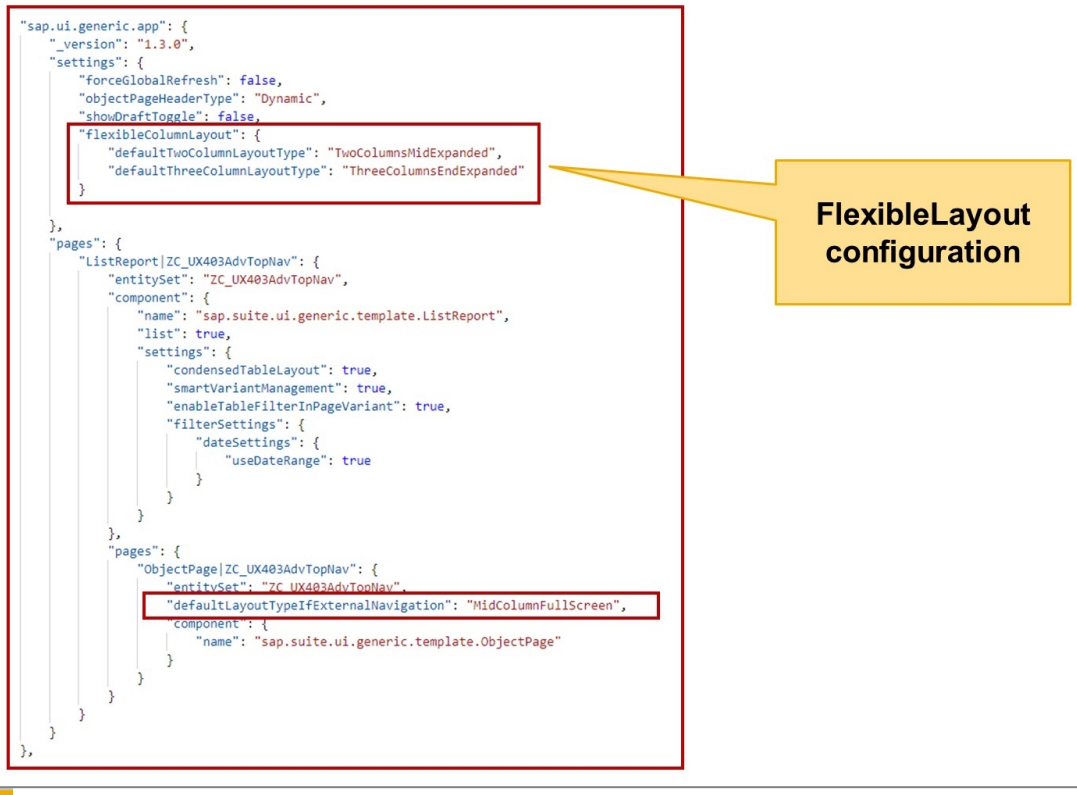


Figure 126: Flexible Layout Configuration (1)

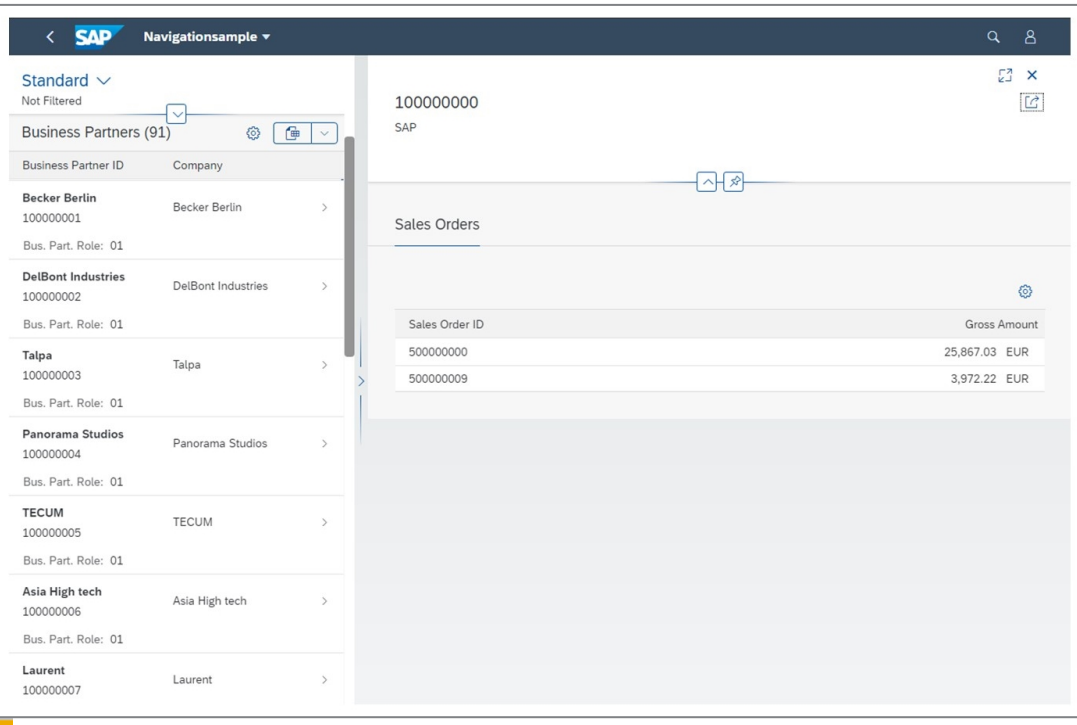


Figure 127: Flexible Layout Configuration (2)

© Copyright. All rights reserved.

Options of External Navigation

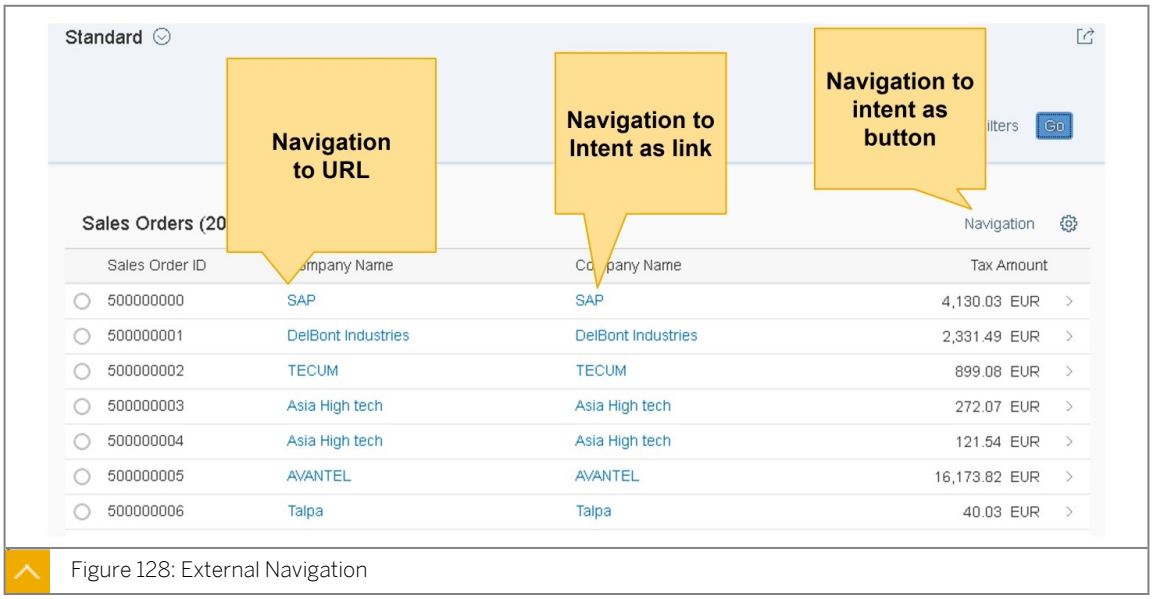


Figure 128: External Navigation

In the SAP Fiori elements app, you can navigate to other addresses by using external navigation.

This function works both on the list report and the object page.

When creating external navigation, you can choose from two alternatives:

- Using a URL: Navigates to any URL
- Using Semantic Object navigation

For the semantic object navigation, you can choose between:

- Display as a link.
- Use a button on the toolbar.

Note:
For detailed information of semantic object and intent based navigation, please refer to UX100.

© Copyright. All rights reserved.

Figure 129: External Navigation-Using a URL

To create a external navigation using a URL, you need to do the following:

- Create a field represent as URL.
- For the field, add a link, set the type to *#WITH_URL*, and point to the URL field using the *url* property.

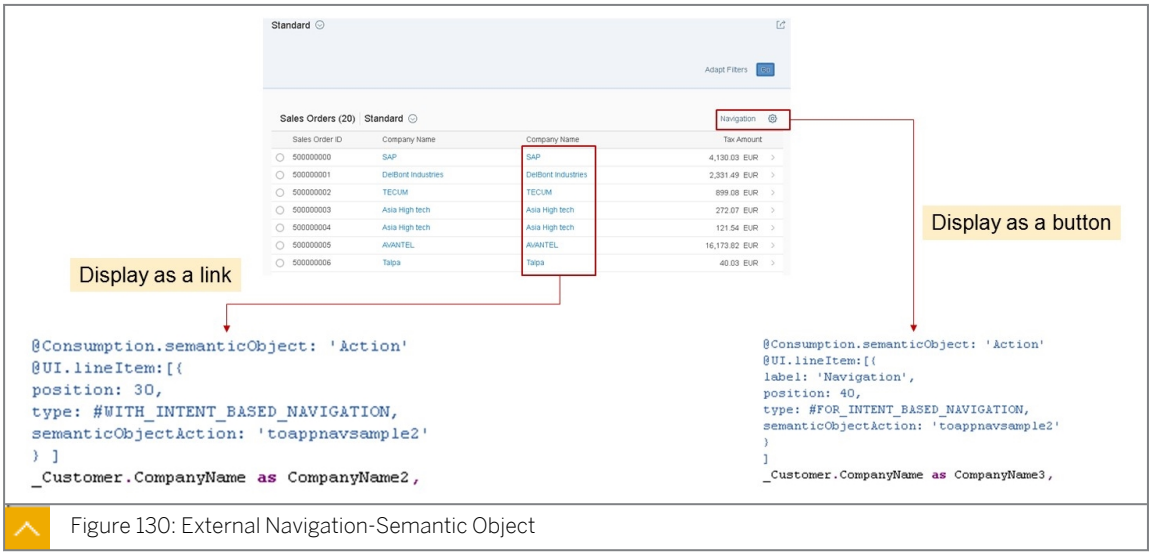


Figure 130: External Navigation-Semantic Object

For navigation to semantic objects, you need to do the following:

- Add a *@consumption.semanticObject* before the field.
- Set *UI.lineItem.type* to **#WITH_INTENT_BASED_NAVIGATION** (as a link) or **#FOR_INTENT_BASED_NAVIGATION** (as a button).
- Set *@UI.lineItem.semanticObjectAction*.

The generated URL contains the semantic object and action. All values in the entity pass as parameters.

Here is an example:

https://.....#Product-display?CompanyName=SAP&CompanyName2=SAP&
Currency=EUR&NetAmount=21737.00&SalesOrder=500000000&TaxAmount=4130.03

© Copyright. All rights reserved.

LESSON SUMMARY

You should now be able to:

- Explain the navigation concept and annotations
- Describe options of external navigation

© Copyright. All rights reserved.

© Copyright. All rights reserved.

Using Data Visualization

LESSON OBJECTIVES
After completing this lesson, you will be able to:

- Use data visualization

Data Visualization

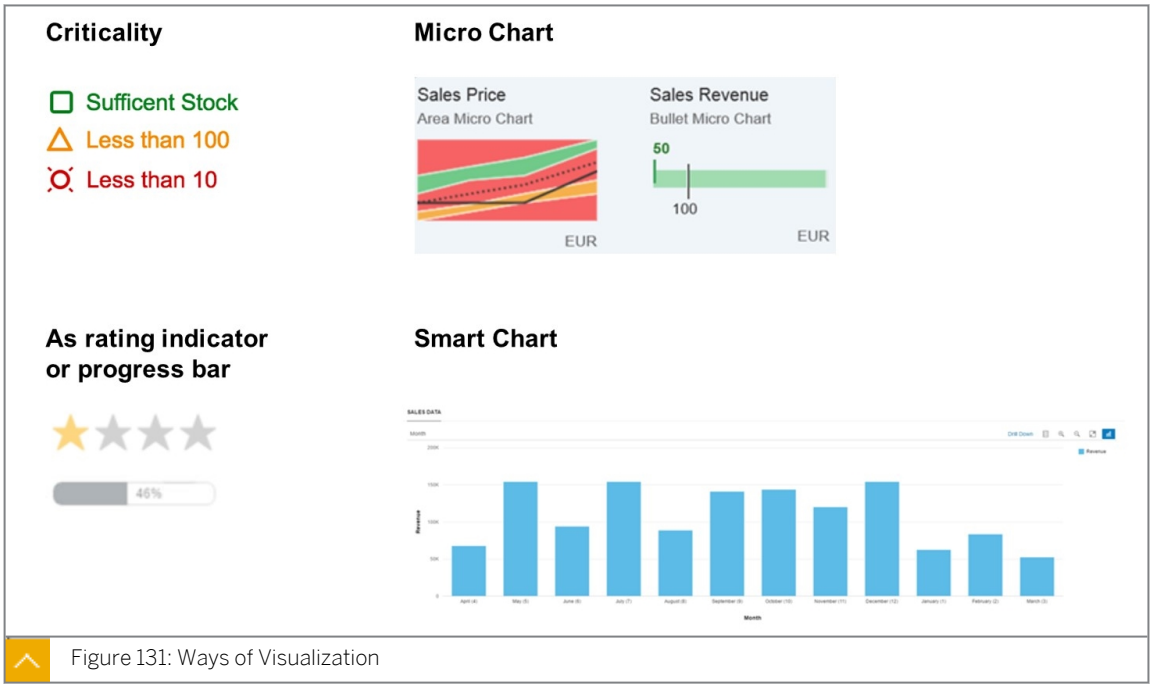


Figure 131: Ways of Visualization

There are several methods to visualize your data in a list report and on an object page.

To display a value in a different state (Neutral, Negative, Critical, Positive) with different colors, you can use the concept of criticality to visualize it.

For data to represent a rating or progress, a rating indicator and a progress bar can be used to display them in a more vivid way.

For the header of an Object Page, you can add an area micro chart or a bullet micro chart as a header facet to display the relationships between actual value, target value, predictive value, max value, and min value.

In the section area of the object page, you can add a smart chart as a section to get insight from an entity set with 0..* association.

© Copyright. All rights reserved.

Criticality

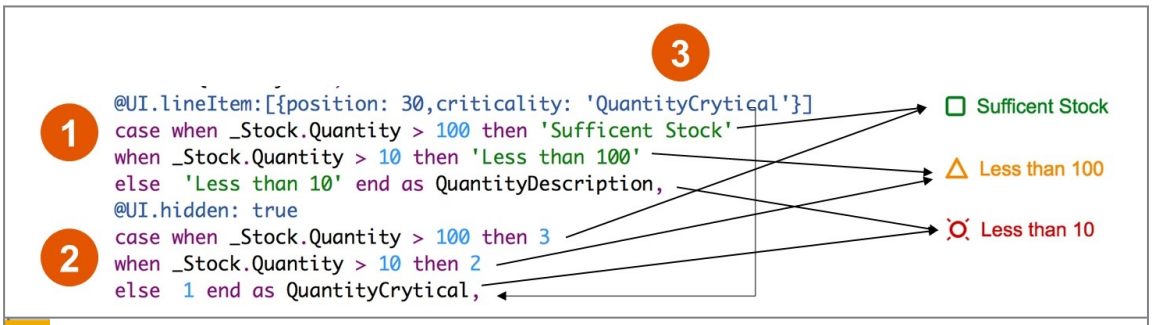


Figure 132: Criticality

Criticality can be used in any data field displayed on a list report and an object page. It also works with other annotations with DataField as datatype like UI.fieldGroup and UI.Identification.

If you are using local annotation, criticality is a optional attribute for DataField, you can either set it to a static value or reference it to a property.

Steps to use criticality in list report by writing CDS view are:

1. You have a *Data Field* to display some text as *Critically Description Field*, the output may be different for each criticality status.
2. Create another integer *Data Field* as *Critically Value Field* having values between 0 and 3, the meanings of the values are:
   - 0 - Neutral
   - 1 - Negative
   - 2 - Critical
   - 3 - Positive

   Normally, this field is calculated based on quantity or status fields.

   To prevent users displaying the number directly, the field should be set to hidden.
3. Writing Annotation for Criticality Description Field were been created in a previous step. Point critical information to the Critically Value Field, using criticality property of @UI.lineitem.

   The following are default colors for status:
   - Neutral = Black
   - Negative = Red
   - Critical = Yellow
   - Positive = Green

© Copyright. All rights reserved.

Note:
You can use the Theme Designer to change these default colors by adjusting the theme. For detailed information for Theme Designer, refer to UX100.

Rating Indicator and Progress Bar

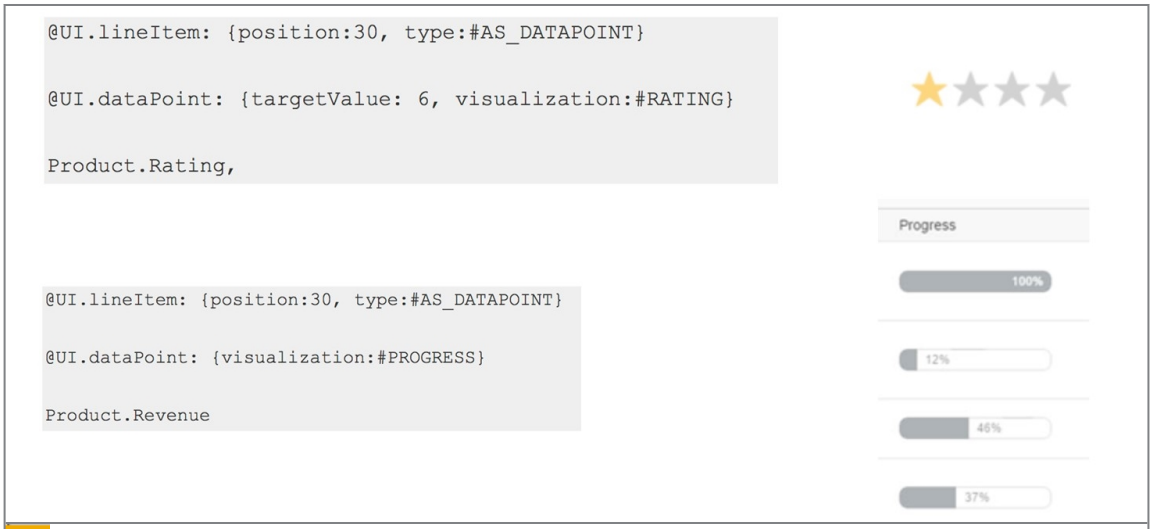


Figure 133: Rating Indicator and Progress Bar

For rating and progress data, you can use the rating indicator and the progress bar.

To use those controls, set your @UI.lineItem (also apply to @UI.fieldGroup , @UI.identification, and other annotations refer to a DataField) to the type of #AS_DATAPOINT. Then, add a @UI.dataPoint annotation and set visualization property to @RATING or #PROGRESS.

By default, the rating indicator has five stars and the progress bar is set to 100 as the target. You can customize those further by setting the value of the *targetValue* property.

LESSON SUMMARY

You should now be able to:

- Use data visualization

© Copyright. All rights reserved.

© Copyright. All rights reserved.

Unit 5
Lesson 3

Creating Charts

LESSON OBJECTIVES

After completing this lesson, you will be able to:

- Create Charts

Charts

OData Service with Analytic Function

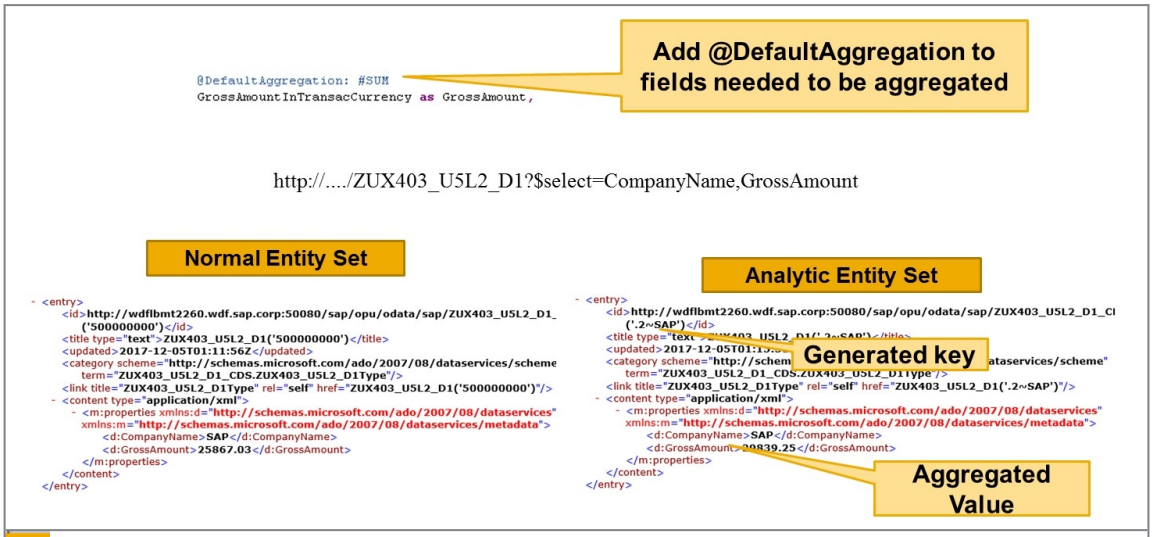


Figure 134: OData Service with Analytic Function

A smart chart control sends an aggregate request to the back end service by sending the *$select* OData parameter.

A normal entity set will return all records only with fields in *$select*, but an analytic entity set will return an aggregated result considering the fields in *$select*.

To define an analytic entity set, add a *@defaultAggregation* annotation to fields you need to aggregate (normally are amount or quantity fields).

The generated entity set has the following differences compared to a normal entity set:

- A generated ID field is used as primary key instead of key of the CDS view.
- When accessed by $select, the engine tries to determine if the fields in $select are attributes or measures according to *@defaultAggregation*, then calculate the aggregated measures grouped by attributes in *$select*.

A typical scenario is to define the CDS view for the list report and the object page as a normal entity set and define that the CDS view represents detailed level entry as an analytic entity

© Copyright. All rights reserved.

set. Then, create a 1 to * association from master level entity to detail level entity. Then, the user can get summarized detail data from the object page.

> Note:
> Another method of creating an analytic OData service is Analytic Query, for detailed information of Analytic Query, refer to course S4H410.

Add @UI.Chart Annotation

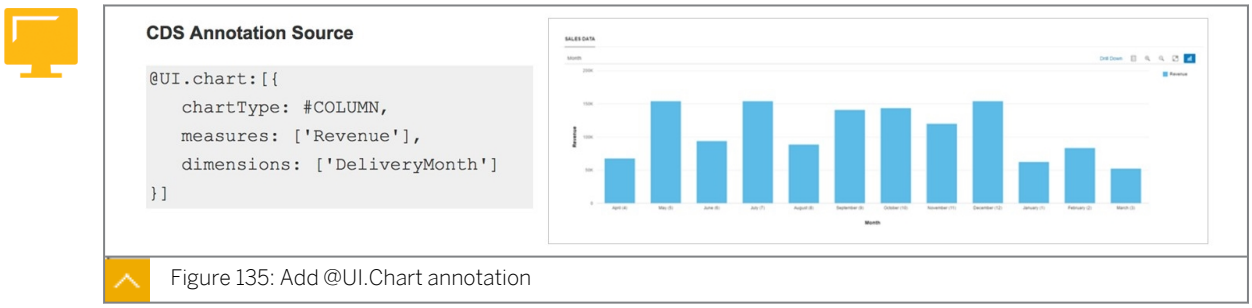


Figure 135: Add @UI.Chart annotation

A *@UI.Chart* annotation is used to define a smart chart. In the *chartType* property, you can specify the type of chart. You need to specify measures and dimensions.

The *@UI.Chart* term is also available in local annotation. Normally a *@UI.Chart* is defined in a analytic CDS view.

Adding a Reference Facet

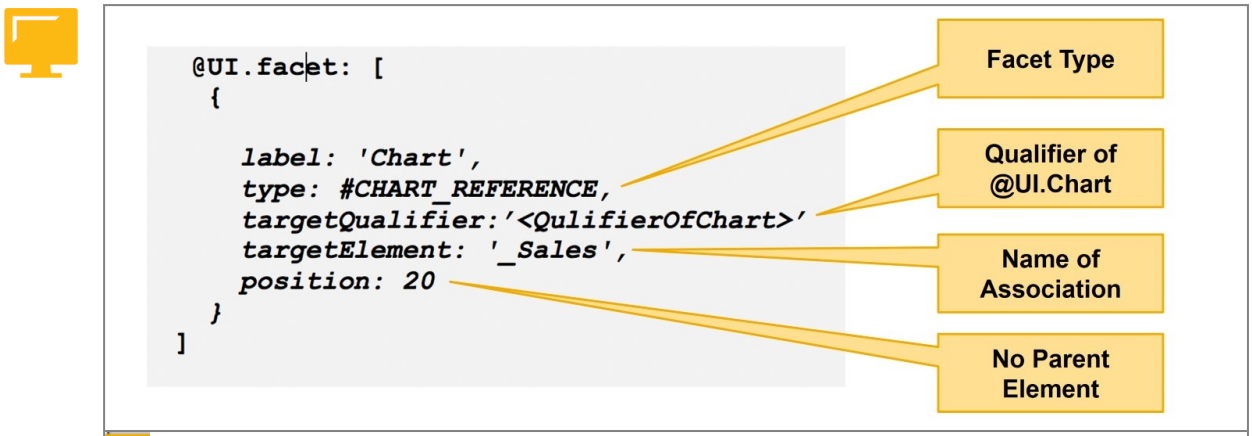


Figure 136: Adding a Reference Facet

A reference facet should be created as a direct child of *@UI.facet* with type *#CHART_REFERENCE* and reference to *@UI.Chart* annotation through an association.

© Copyright. All rights reserved.

Micro Bullet Chart

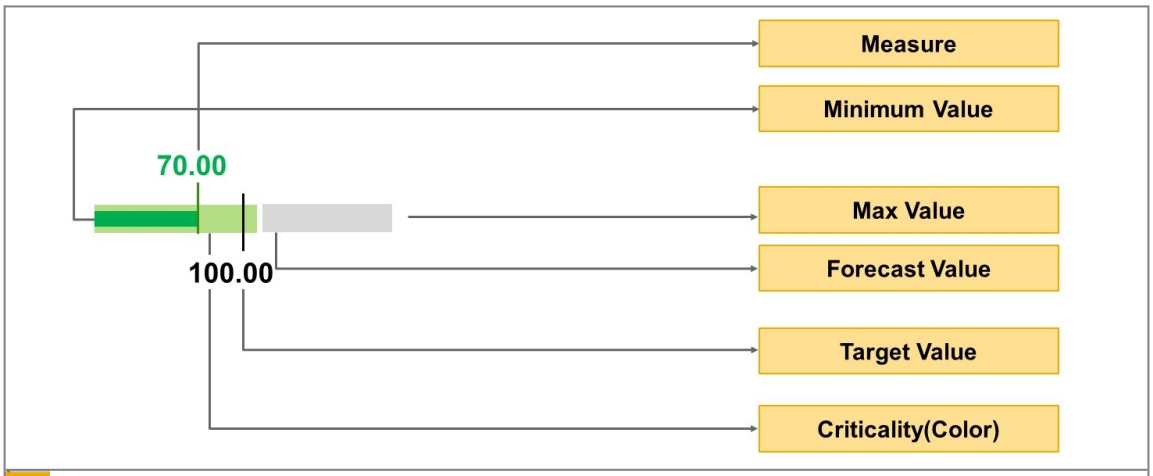


Figure 137: Micro Bullet Chart

A Micro Bullet chart is used to evaluate a key figure by entering its actual value, minimum value, max value, forecast value, and target value together, and then showing the criticality by the color. It's useful to get insight among target, plan, and actual, and widely used in analytic scenarios.

The bullet chart can be used as a header facet in the object page.

Example of a Bullet Chart

```
@UI.chart:[{
  title:'Sales Revenue',
  description: 'Bullet Micro Chart',
  chartType: #BULLET,
  measures:['BulletChartRevenue'],
  measureAttributes: [
    { measure: 'BulletChartRevenue', role: #AXIS_1, asDataPoint: true }
  ]
}]
```

```
@UI.dataPoint: {
   title:'Sales Revenue',
   targetValueElement: 'TargetRevenue',
   forecastValue: 'ForecastRevenue',
   minimumValue: 100,
   maximumValue: 300,
   criticality: 'Criticality'
}
ProductSalesRevenue.Revenue as BulletChartRevenue
```

Figure 138: Example of a Bullet Chart

Like smart charts, the creation of micro chart involves *@UI.facet* and *@UI.chart*.

In contrast to smart charts, a micro chart has more detailed insight to a point of data. So that a *@UI.dataPoint* is involved in the creation of a micro chart as *@UI.chart* does not contain all of the detailed information for a point of data. The evaluated value should be the only member in the *@UI.chart.measure* and it must link to a *@UI.datapoint* annotation through *measureAttributes*. All other values (max,min,target,forecast ...) are assigned in *@UI.dataPoint*.

© Copyright. All rights reserved.

To Create a Chart

1. Create an OData entity set with analytic function.
2. Create @UI.Chart annotation on the analytic OData entity set.
3. Create a Reference Facet to display the chart.

LESSON SUMMARY

You should now be able to:

- Create Charts

© Copyright. All rights reserved.

Performing CRUD operations with BOPF

LESSON OBJECTIVES

After completing this lesson, you will be able to:

- Perform CRUD operations with BOPF

Business Object Processing Framework (BOPF)

Create and Delete Data with SAP Fiori Elements

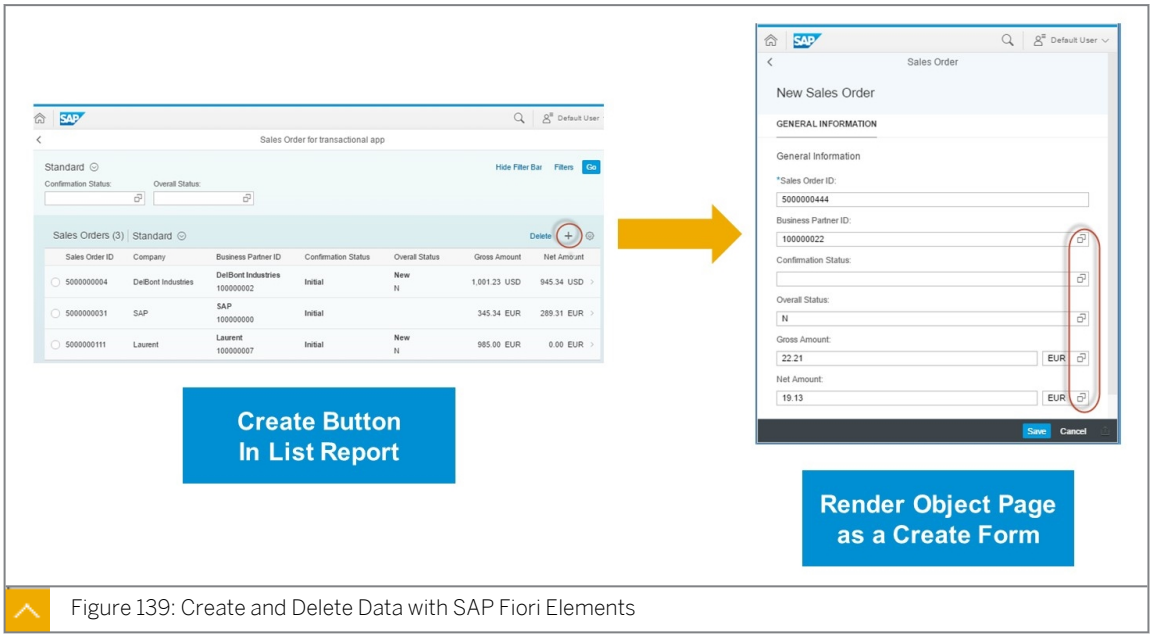


Figure 139: Create and Delete Data with SAP Fiori Elements

You can use the List Report and an Object to create a new object.

By adding a + button to the toolbar, you can navigate to the object page where all fields are editable to create a new object.

You can also delete a record by selecting the radio button on the left of the record and then choosing *Delete* on the toolbar.

© Copyright. All rights reserved.

Updating a Record

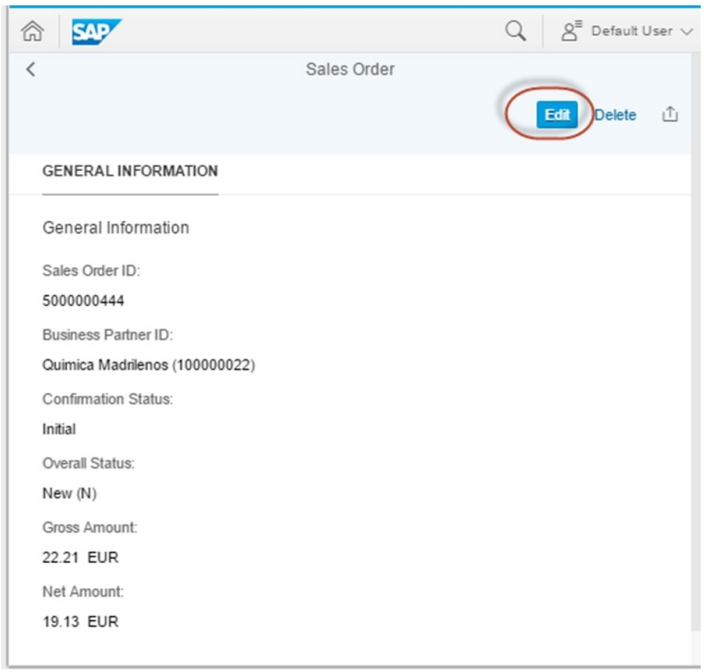


Figure 140: Update a Record using SAP Fiori Elements

When you are on an object page, you can use the *Edit* button to switch between display mode and edit mode. You can also delete this record by choosing *Delete*.

Draft Control

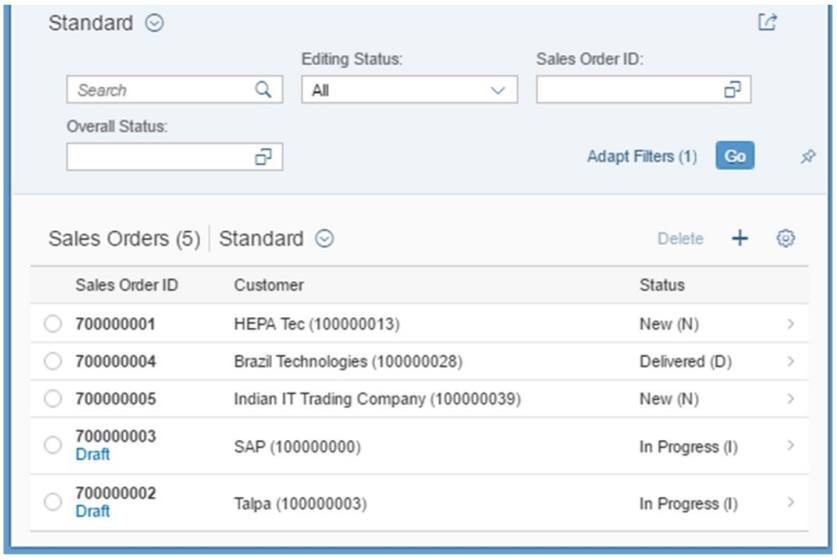


Figure 141: Draft Control When Creating or Changing a Record

SAP Fiori elements can also handle the status of a record.

For example, when User A wants to update a record, a draft is created for this record. If User B wants to update this record at the same time, his account is blocked from making the change.

© Copyright. All rights reserved.

Architecture

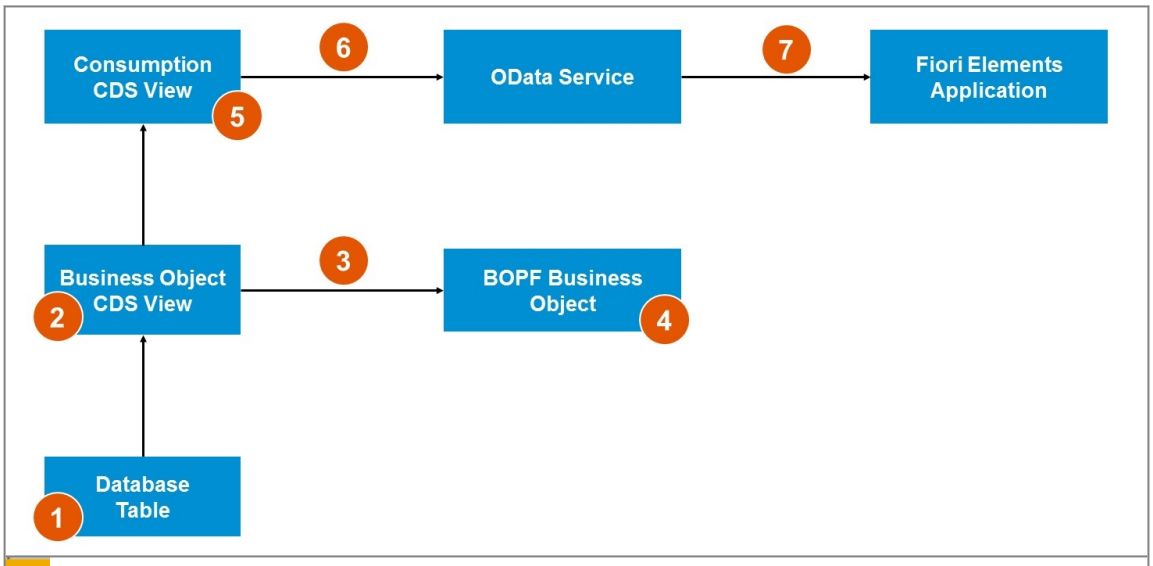


Figure 142: Architecture of an SAP Fiori Elements Application with Write Capability

The figure, Architecture of an SAP Fiori Elements Application with Write Capability, shows the architecture for an SAP Fiori elements application with write capability:

1. Create a database table.

   A database table is needed to store records. The table must include structure "/bobf/s_lib_admin_data".

2. Create a business object CDS view.

   Create a CDS view which selects data from the table you've created and then create associations to connect with relative business entities.

3. Generate a BOPF business object.

   In BOPF, a business object is represented as a hierarchical tree of nodes. A single node includes a set of semantically related attributes and the corresponding business logic. For each node, several types of entities can be defined to describe the specific business logic part of the business object.

   You can generate BOPF by adding annotations in the Business Object CDS View.

4. Add additional business logic in BOPF.

   Additional business logic written in ABAP code can be added to BOPF. For example, you can add a validation code which can be executed before the record is saved in the database.

5. Create a consumption CDS view and write UI annotation on it.

   Start with your CDS view for the SAP Fiori elements application and write UI annotation on it.

6. Generate OData Service.

   Perform this step as you've learned in this course.

7. Generate SAP Fiori Elements application.

   Perform this step as you've learned in this course.

© Copyright. All rights reserved.

Note:
For Detailed information of how to create transactional list report and object page.

Please proceed as follows:

Open URL https://help.sap.com/viewer/p/SAP_NETWEAVER_AS_ABAP_752, then navigate to Development→Development Information→Application Development on AS ABAP→SAP - ABAP Programming Model for SAP Fiori→Develop→Developing New Transactional Apps

LESSON SUMMARY

You should now be able to:

- Perform CRUD operations with BOPF

© Copyright. All rights reserved.

Learning Assessment

1. What are the options for external navigation?

   Choose the correct answers.

   - ☐ **A** Navigation to URL (As a link).
   - ☐ **B** Navigation to URL (As a button).
   - ☐ **C** Navigation to Intent (As a link).
   - ☐ **D** Navigation to Intent (As a button).

2. What can you do by editing *manifest.json*?

   Choose the correct answers.

   - ☐ **A** Disable object page.
   - ☐ **B** Add object pages as sub page under an object page.
   - ☐ **C** Define facets on Object Page.
   - ☐ **D** Disable List Report.

3. The field indicate criticality should be hidden because it means nothing to end user.

   Determine whether this statement is true or false.

   - ☐ True
   - ☐ False

4. What steps are used to display a field as a rating indicator?

   Choose the correct answers.

   - ☐ **A** Set type of line item to *#AS_DATAPOINT*.
   - ☐ **B** Add a *@UI.datapoint* to the field and set visualization to *#RATING*.
   - ☐ **C** Add a *@UI.chart* annotation to the field and set visualization to *#RATING*.
   - ☐ **D** Set the visualization of line item to *#RATING*.

© Copyright. All rights reserved.

5. What are the differences between analytical entity set and normal entity set from an OData consumer perspective?

   Choose the correct answers.

   ☐ A The Analytical entity set generates a new field for primary key.

   ☐ B The Analytical entity set returns all data needed to analyze the client.

   ☐ C The Analytical entity set returns summarized results according to the *$select* parameter.

   ☐ D The Analytical entity set analyzes the use of the database and runs faster when SAP HANA is used as database compared to a normal entity set.

6. SAP Fiori elements support write operations, if the back end service is written by CDS working with BOPF.

   Determine whether this statement is true or false.

   ☐ True

   ☐ False

© Copyright. All rights reserved.

Learning Assessment - Answers

1. What are the options for external navigation?

   Choose the correct answers.

   - [X] A Navigation to URL (As a link).
   - [ ] B Navigation to URL (As a button).
   - [X] C Navigation to Intent (As a link).
   - [X] D Navigation to Intent (As a button).

   Correct. The options for external navigation are: navigation to URL (as a link), navigation to intent (as a link), and navigation to intent (as a button).

2. What can you do by editing *manifest.json*?

   Choose the correct answers.

   - [X] A Disable object page.
   - [X] B Add object pages as sub page under an object page.
   - [ ] C Define facets on Object Page.
   - [ ] D Disable List Report.

   Correct. By editing *manifest.json* you can disable object page and add object pages as sub page under an object page.

3. The field indicate criticality should be hidden because it means nothing to end user.

   Determine whether this statement is true or false.

   - [X] True
   - [ ] False

   Correct. The field indicate criticality should be hidden because it means nothing to end user.

© Copyright. All rights reserved.

4. What steps are used to display a field as a rating indicator?

   Choose the correct answers.

   - [X] A Set type of line item to *#AS_DATAPOINT*.
   - [X] B Add a *@UI.datapoint* to the field and set visualization to *#RATING*.
   - [ ] C Add a *@UI.chart* annotation to the field and set visualization to *#RATING*.
   - [ ] D Set the visualization of line item to *#RATING*.

   Correct. To display a field as a rating indicator, set type of line item to *#AS_DATAPOINT* and add a *@UI.datapoint* to the field and set visualization to *#RATING*.

5. What are the differences between analytical entity set and normal entity set from an OData consumer perspective?

   Choose the correct answers.

   - [X] A The Analytical entity set generates a new field for primary key.
   - [X] B The Analytical entity set returns all data needed to analyze the client.
   - [ ] C The Analytical entity set returns summarized results according to the *$select* parameter.
   - [X] D The Analytical entity set analyzes the use of the database and runs faster when SAP HANA is used as database compared to a normal entity set.

   Correct. The differences are: The Analytical entity set generates a new field for primary key. The Analytical entity set returns all data, needed to analyze the client. The Analytical entity set analyzes the use of the database and runs faster when SAP HANA is used as database compared to a normal entity set.

6. SAP Fiori elements support write operations, if the back end service is written by CDS working with BOPF.

   Determine whether this statement is true or false.

   - [X] True
   - [ ] False

   Correct. SAP Fiori elements support write operations, if the back end service is written by CDS working with BOPF.

© Copyright. All rights reserved.

UNIT 6 Overview Page

Lesson 1

UNIT OBJECTIVES

- Get an overview of the Overview Page (OVP)

© Copyright. All rights reserved.

© Copyright. All rights reserved.

Unit 6
Lesson 1

Getting an Overview of the Overview Page (OVP)

LESSON OBJECTIVES

After completing this lesson, you will be able to:

- Get an overview of the Overview Page (OVP)

Overview of the Overview Page (OVP)

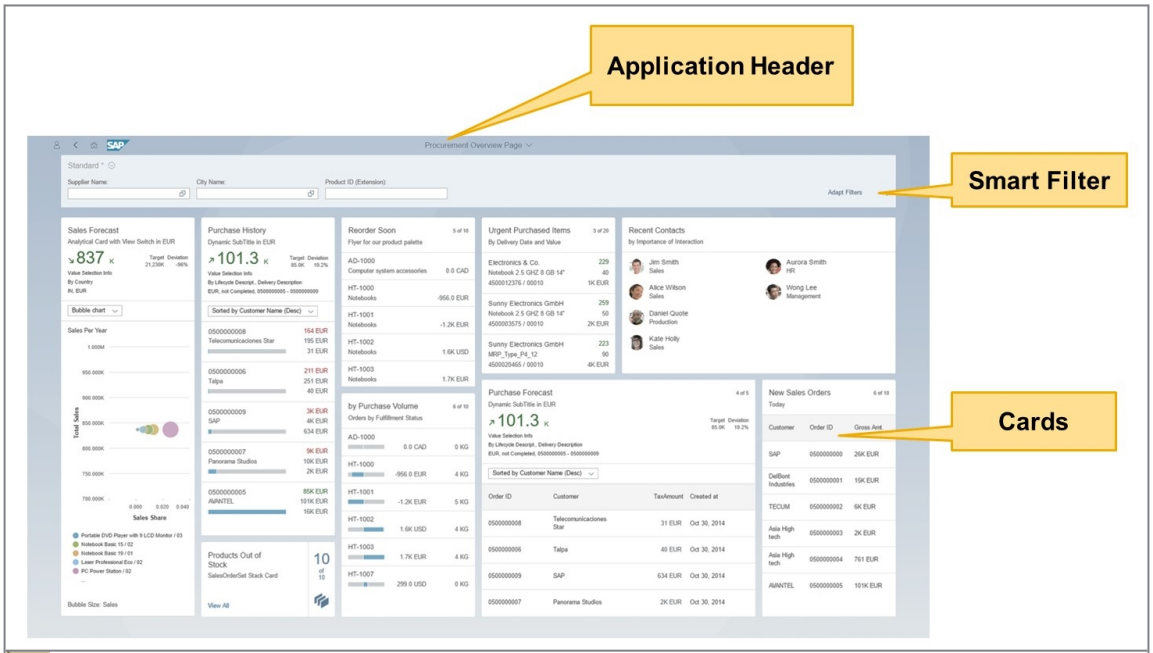


Figure 143: Overview Page

Overview pages provide quick access to vital business information at a glance, in the form of visual, actionable cards. The user-friendly experience makes viewing, filtering, and acting upon data quick and simple. While simultaneously presenting the big picture at a glance, business users can focus on the most important tasks enabling faster decision making as well as immediate action.

The application lets you create several cards for different types of content that helps in visualizing information in an attractive and efficient way. You can create overview pages and add cards to the page using the overview page wizard in SAP Business Application Studio.

The displayed data is fully interactive, with clickable areas for easy navigation to relevant applications. Based on SAP Fiori, overview pages organize action items with a fully responsive user interface. Users can access overview pages from SAP Fiori launchpad and narrow down the information displayed, or opt to hide cards to focus on a particular topic.

© Copyright. All rights reserved.

Main Components of the Overview Page Application

The overview page application contains the following main components:

- Application header: Provides a description of the area for which this application provides an overview (for example, procurement or sales). From the header area, users can change user account settings and manage cards (show/hide).
- Smart filter: Provides application-level filters for changing the levels of data displayed in the cards. For example, you could use the filter to display only transactions larger than $10,000, only items lighter than 50 kg, and so on.
- Cards: A card is a smart component that uses UI annotation to render its content. Each card is bound to a single entity set in a data source. A card may display a donut or bar chart, or a table. Stack cards contain a set of quick view cards which can be viewed in an object stream. Cards are displayed on the overview page in up to five responsive columns and can be rearranged by dragging and dropping.

Overview page application instances consist of a UI component that extends the overview page application component and a manifest file that contains the application configuration.

Types of Cards

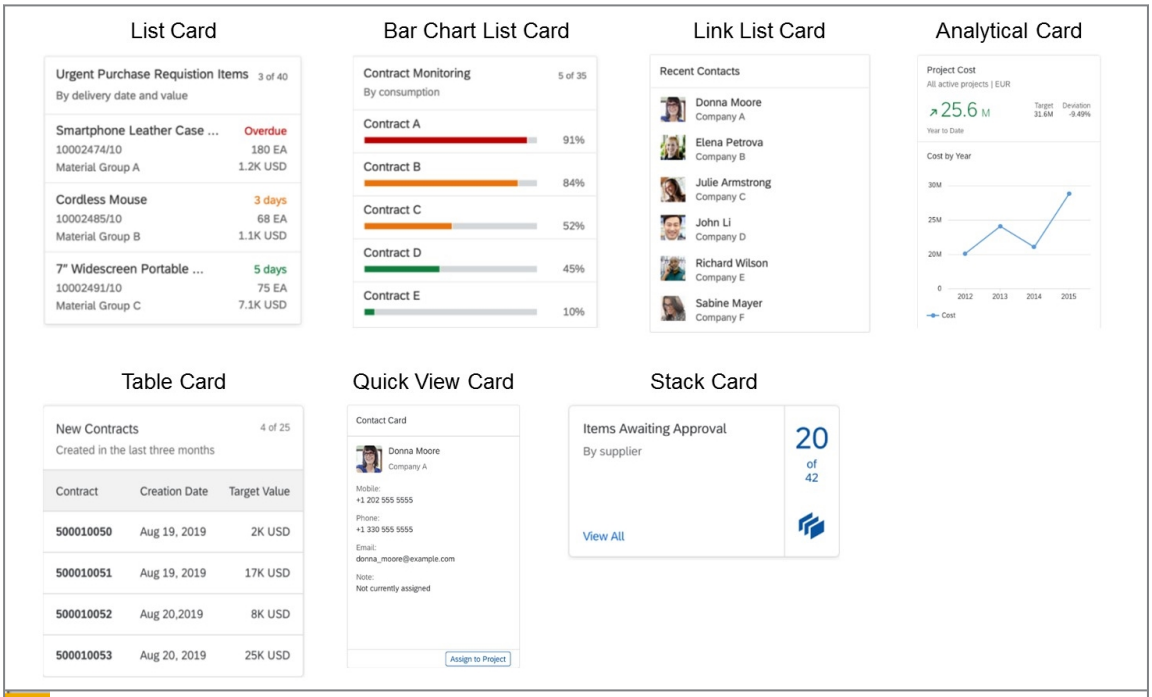


Figure 144: Types of Cards

The following are explanations about the various types of cards:

List Cards

List cards display lists of records according to the configuration in the com.sap.vocabularies.UI.v1.LineItem term. List cards display up to six fields of data in each list item.

Bar Chart List Card

© Copyright. All rights reserved.

Bar chart list cards are a type of object group card, and display a set of items in a vertical list. Unlike list cards, bar chart list cards are embedded in another component: the comparison micro chart.

Link List Cards

Allows you to view a list of links with title, picture, icon, or subtitle.

Analytic Cards

Analytical cards lets you view data in a variety of chart formats. The card is divided into two areas (header and chart).

Table Cards

A table card displays a list of records according to the configuration in the com.sap.vocabularies.UI.v1.LineItem term. A table card displays data in a 3-column table layout. (Optional) You can configure smart links in table cards to access quick links.

Quick View Cards

Quick view cards are single-object cards. They display the basic details for one object, such as the name, address, and phone numbers for a contact.

Stack Cards

Stack cards aggregate a set of cards of the same type, which are based on a common topic or action. When selected, up to 20 stacked cards can be displayed in the object stream.

Custom Cards

Custom cards allow you to define the appearance of a card on an overview page, and the type of content that appears in the card content area. They offer additional flexibility when you require features that are not offered by the standard cards for overview pages.

Note:
For more information of type of cards, please access website: https://sapui5.netweaver.ondemand.com/sdk/#/topic and navigate to *Developing Apps with SAP Fiori Elements → Overview Pages → Overview Page Card → Types of Cards*.

To Create an Overview Page

1. Prepare the back end service for a global filter.
2. Prepare the back end service for cards.
3. Create an SAP Fiori application using the Overview Page template.
4. Add cards for the Overview Page.

© Copyright. All rights reserved.

Overview Page Creation

1st Step

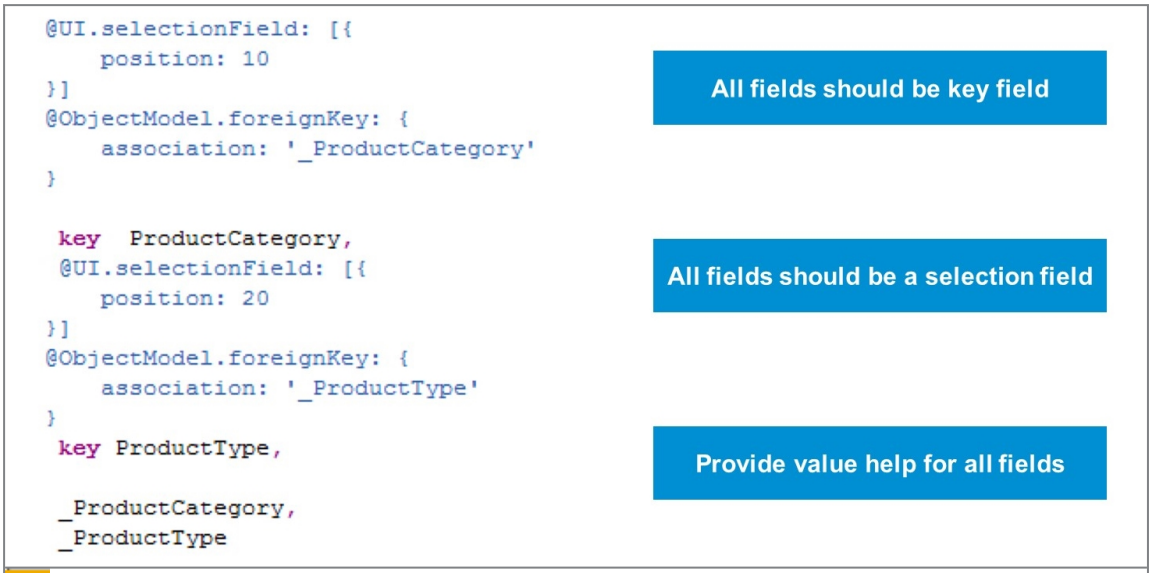


Figure 145: 1. Prepare Back end Service for Global Filter

For most overview pages, there is an OData service supporting the global filter on the top of the page.

The OData service for global filter should only contain attribute values and should expose these as key fields. All fields should be exposed as a selection field. A value help created either by foreign key association or modeled value help should be assigned to the selection field.

2nd Step

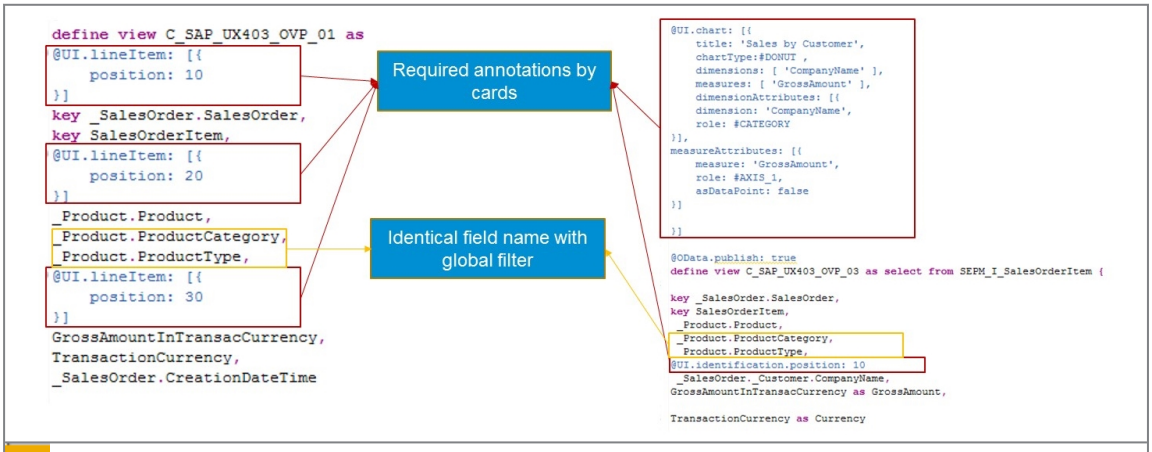


Figure 146: 2. Prepare Back end Services for Cards

You can provide a back end service in the same OData service of the global filter or in a standalone OData service.

Generally, for a global filter and each card in an overview page, you need a group of CDS views (entity sets) to server the main query as well as value help or other associated data access. Cards in an overview page usually involve lots of different base CDS views, so in most cases, we create a separate OData service for each card, except for cards that have the same back end logic.

© Copyright. All rights reserved.

Different type of cards require different annotations on a CDS view. For example, table cards require `@UI.lineItem`, analytic cards require `@UI.Chart` and `@UI.identification`. For detailed information, please refer to detailed information for card types.

Cards and global filters are connected by fields, so, if you want to filter data on a card by setting a global filter, the cards need to have fields with identical name fields in a global filter. If the names don't match, the filter is ignored.

3rd Step

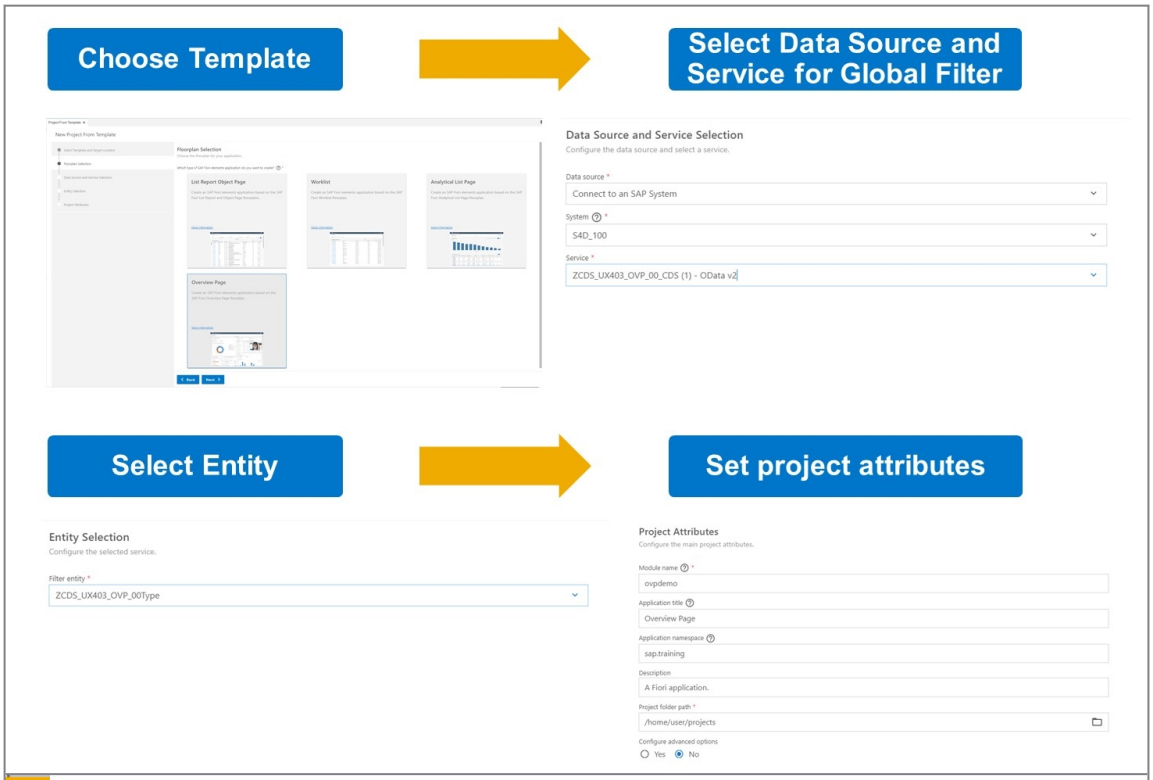


Figure 147: 3. Create an SAP Fiori application using Overview Page template

The process of creating an Overview Page is similar to a list report, as follows:

1. Select an SAP Fiori elements application template from the project templates.
2. Choose Overview Page as floorplan.

4th Step

To add a card to an overview page, you need to do the following:

1. Start *SAP Fiori Tools – Guided Development*.
2. Select the appropriate guide for your card type.
3. Configure the template.

OVP Configuration

- `smartVariantRequired` Enable/Disable the smart variant management control on the smart filter bar.
- `showDateInRelativeFormat` Show date on the SFB like *yesterday*, *tomorrow*, and so on.

© Copyright. All rights reserved.

- `enableLiveFilter` show/do not show go button on the smart filter bar.
- `refreshIntervalInMinutes` refresh the OVP every x minutes, minimum is 1 min.
- `useDateRangeType` : true // true to enable Semantic Date, default false.

Topic: `useBatch`: false (https://sapui5.hana.ondemand.com/#/api/sap.ui.model.odata.v2.ODataModel/constructor)

Consider a scenario where you have one analytical card and four transactional cards (list, table, stack, linklist) in your OVP application. The HTTPS requests made by the analytical cards are generally heavy (take a long time). Since analytical information is based on aggregated entity sets, the back end requires time to process each http request. If all the cards in your application use the same model (datasource), then in such a scenario, there would be only one batch request for all the cards to fetch data. Since the calls for analytical cards take too long, the other four cards would also not display any data. This scenario can be handled by creating two separate models in the application manifest and disabling batch mode for the analytical card model. In this case, the response from the back end would be quicker in and you would see the OVP being rendered relatively quickly.

`preload`: true (https://blogs.sap.com/2016/11/19/sapui5-application-startup-performance-advanced-topics/ - Section: Use model preload feature)

This is a way to load the metadata and annotations before the OVP application load, so that OVP does not have to 'wait' for the metadata and annotations and can continue directly with rendering the UI.

General performance guidelines: if you have analytical cards that take a long time to return data, you can consider disabling the batch mode (the rationale being that multiple small requests fired almost in parallel time would be faster when compared to one heavy batch call). Ideally, the suggested way is to have one model in the manifest which all the analytical cards would use and one model in the manifest which all the transactional cards would use.

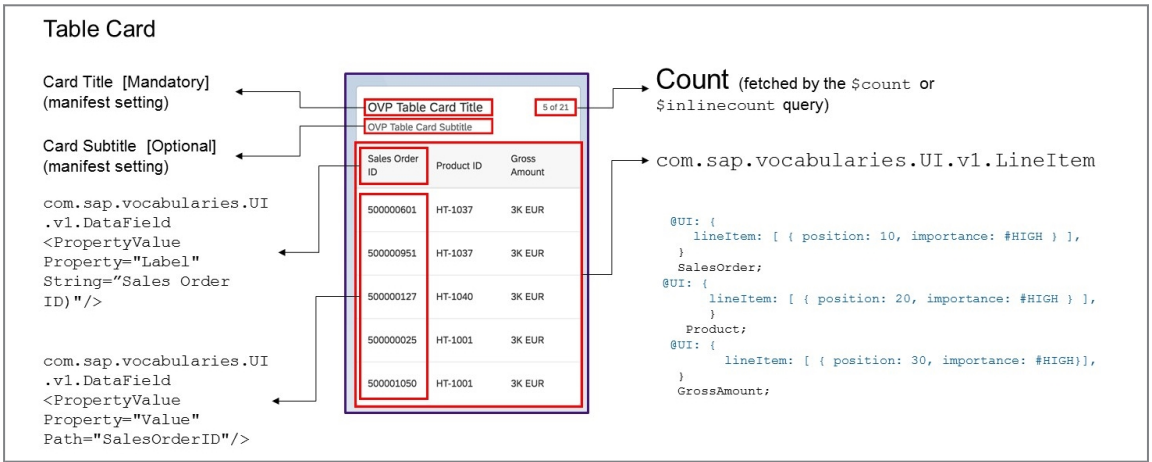


Figure 148: Table Card

© Copyright. All rights reserved.

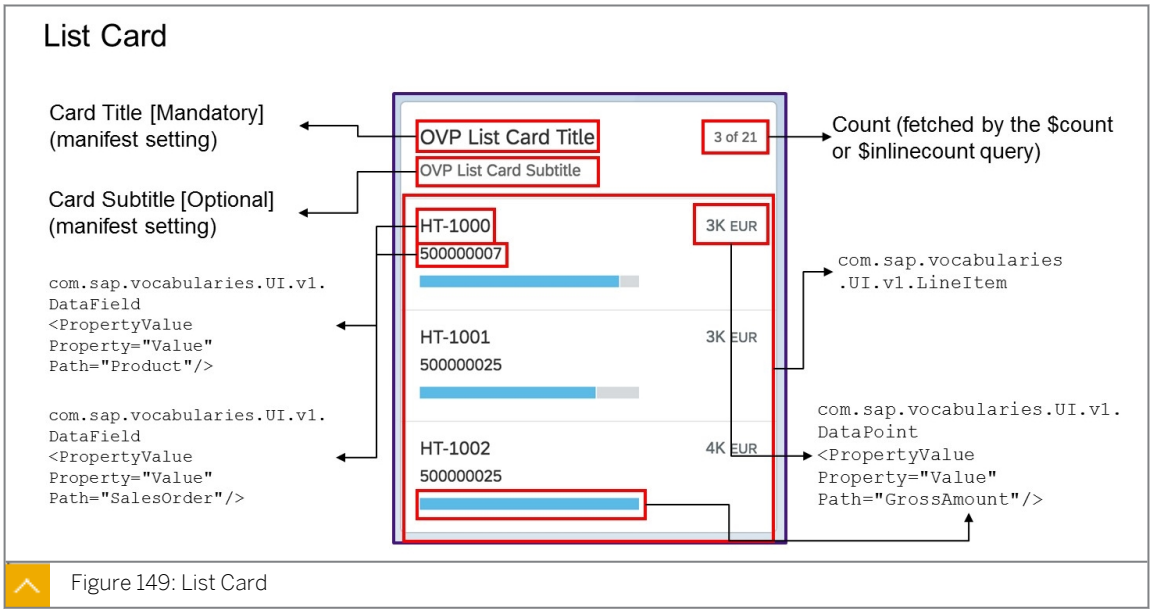


Figure 149: List Card

All the data fields are displayed at the left-hand side. All the data points are displayed at the right-hand side.

For now, we will have to use the local annotation approach because the `datafieldForAnnotation` annotation is not yet supported by CDS.

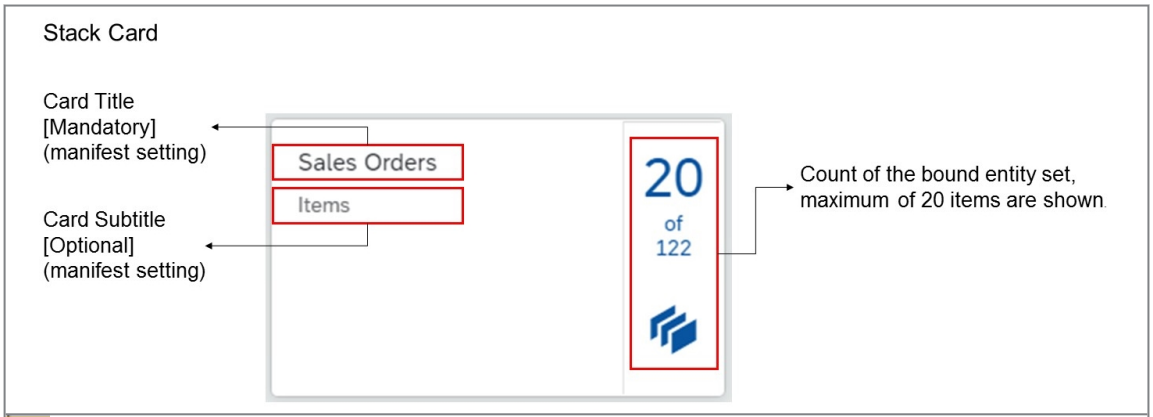


Figure 150: Stack Card

The card configuration for stack cards looks like the following:

```
"card02": {
  "model": "mainService",
  "template": "sap.ovp.cards.stack",
  "settings": {
    "title": "{{card02_title}}",
    "subTitle": "{{card02_subtitle}}",
    "entitySet": "ZCDS_UX403_OVP_00",
    "annotationPath": "com.sap.vocabularies.UI.v1.FieldGroup#DETAILED"
  }
  }
```

The content of the card is implemented as `FieldGroup-Reference-facet`.

© Copyright. All rights reserved.

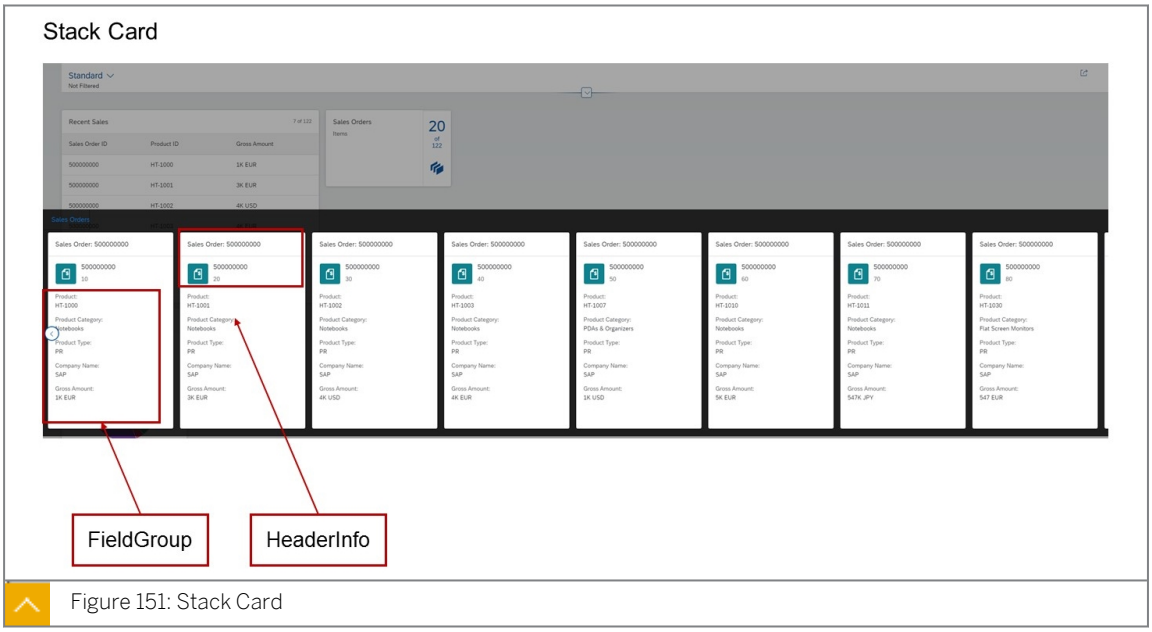


Figure 151: Stack Card

The content of a stack card is implemented by using `headerinfo` and `fieldgroup` annotations

```
@UI.headerInfo: {
typeNamePlural: 'Sales Orders',
typeName: 'Sales Order',
imageUrl: 'SAPIconUrl',
title: {
label: 'Sales Order ID',
value: 'SalesOrder'
},
description: {
label: 'Sales Order Item',
value: 'SalesOrderItem'
}
}
```

Annotate fields of the projection list as fieldgroup items `@UI.fieldGroup:[{position: 20, qualifier: 'DETAILED', label: 'Product Category'}]`.

Create `Fieldgroup-Reference` facet with reference to configured field group id.

```
@UI.facet: [{
type: #FIELDGROUP_REFERENCE,
targetQualifier: 'DETAILED',
isSummary: true
}]
```


© Copyright. All rights reserved.

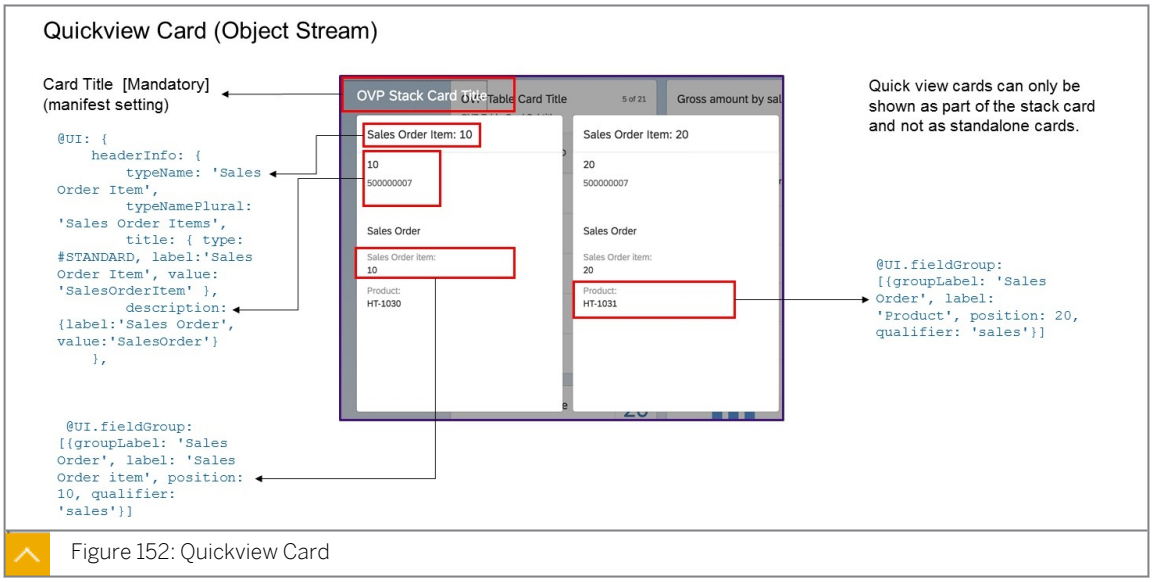


Figure 152: Quickview Card

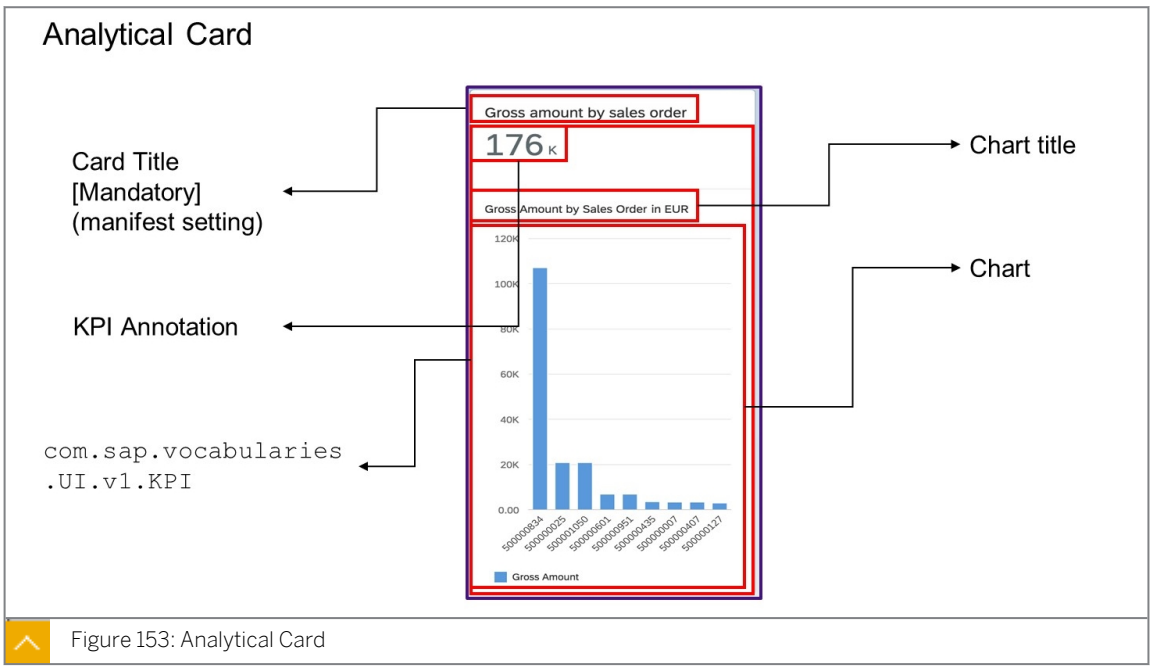


Figure 153: Analytical Card

The manifest.json configuration of an analytical card showing a donut chart looks like the following:

```
"card01": {
  "model": "mainService",
  "template": "sap.ovp.cards.charts.analytical",
  "chartType": "cardchartsdonut",
  "settings": {
    "title": "{{card01_title}}",
    "entitySet": "ZCDS_UX403_OVP_00",
    "identificationAnnotationPath":
"com.sap.vocabularies.UI.v1.Identification",
    "chartAnnotationPath": "com.sap.vocabularies.UI.v1.Chart"
  }
  },
```

© Copyright. All rights reserved.

Figure 154: Static Link List Card

Navigation information is taken from the com.sap.vocabularies.UI.v1.Identification

- `com.sap.vocabularies.UI.v1.DataFieldForIntentBasedNavigation` for intent based navigation within the FLP.
- `com.sap.vocabularies.UI.v1.DataFieldWithUrl` for navigation to external apps and websites.

Figure 155: Navigation from OVP Cards

Documentation: https://help.sap.com/viewer/468a97775123488ab3345a0c48cadd8f/7.52.3/en-US/530f9e6f66104d5888ade79b5cf417e0.html

Note:
The navigation from the card items section is context sensitive, that is, if you select a particular item in the table/list card, this information will be sent to the navigating application as a selection variant.

All the properties related to the particular record will be sent. For example, your table card displays properties P1,P2,P3 as data, when you select the item, all the Odata properties P1,P2,P3....Pn will be sent as a selection variant (will be populated to the smart filter bar). If you would like only P1,P2,P3 to be passed, then use “addOataSelect:true (https://help.sap.com/viewer/468a97775123488ab3345a0c48cadd8f/7.52.3/en-US/530f9e6f66104d5888ade79b5cf417e0.html)

Navigation is possible from all cards and is supported in resizable layout as well.

LESSON SUMMARY
You should now be able to:

- Get an overview of the Overview Page (OVP)

© Copyright. All rights reserved.

Learning Assessment

1. Which of the following types are types of cards in an Overview Page?

   Choose the correct answers.

   - ☐ **A** List Cards
   - ☐ **B** Link List Cards
   - ☐ **C** Analytic Cards
   - ☐ **D** Table Cards
   - ☐ **E** Stack Cards

© Copyright. All rights reserved.

Learning Assessment - Answers

1. Which of the following types are types of cards in an Overview Page?

   Choose the correct answers.

   [X] A List Cards

   [X] B Link List Cards

   [X] C Analytic Cards

   [X] D Table Cards

   [X] E Stack Cards

   Correct. Types of cards in an Overview Page are: List Cards, Link List Cards, Analytic Cards, Table Cards, and Stack Cards.

© Copyright. All rights reserved.

UNIT 7 Analytical List Page

Lesson 1

UNIT OBJECTIVES

- Get an overview of the Analytical List Page

© Copyright. All rights reserved.

© Copyright. All rights reserved.

Unit 7
Lesson 1

Getting an Overview of the Analytical List Page

LESSON OBJECTIVES
After completing this lesson, you will be able to:

- Get an overview of the Analytical List Page

Overview of the Analytical List Page

What is analytical about Analytical List Page (ALP)?

The ALP allows an end user to quickly analyze the data and start focusing on the very records that need attention. It enables this in multiple ways:

- The visual filters enable the user to look at the top (or bottom) x records. This means user can very quickly filter on the top performing cost centers or bottom performing sales centers (something that is not possible using the traditional compact filters).
- The presence of charts in the content area allows the user to quickly assimilate the full picture. A line chart or a bubble chart there can quickly give a lot of information to the end user and the user can further narrow down to the problematic areas by doing one more selection in the chart. This will refresh the table below with only the records that match the chart selection.
- The user can thus very easily ensure that the table in ALP shows only the records of interest. Furthermore, the user can easily select these records for further navigation (to Object Page or custom application page), or take some actions on selected records.

Analytical List Page (ALP) is an SAP Fiori elements application for detailed analytics. It lets you analyze data from different perspectives, to investigate a root cause, and to act on transactional content. You can identify relevant areas within data sets or significant single instances using data visualization and business intelligence. All of this can be done seamlessly within one page. The combination of transactional and analytical data using chart and table visualization lets you to view relevant data more quickly. This hybrid view of data allows an interesting interplay between the chart and table representations.

Configure ALP to include the following use cases seamlessly within one page:

- Related KPIs (key performance indicators) on the header area as KPI tags. These KPI tags further allow a progressive disclosure and navigation through KPI cards.
- Filter data sets used for the main content area through different filter modes. For example, visual filters provide an intuitive way of choosing filter values from an associated measure value.
- Seamless navigation to applications from the content area and KPI card area.
- Customizing and sharing ALP as a page variant with other users.

© Copyright. All rights reserved.

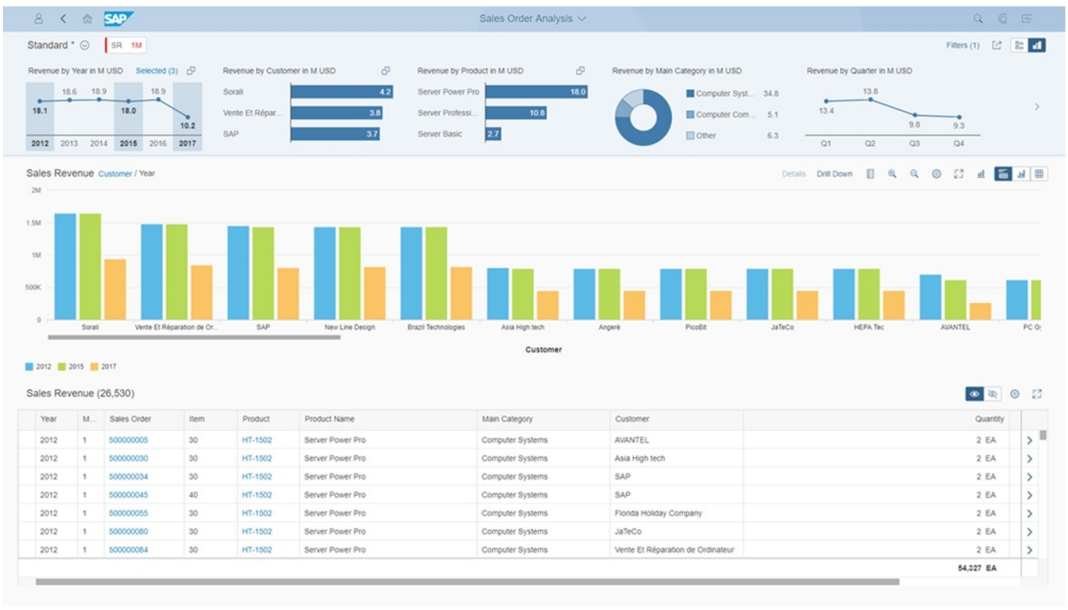


Figure 156: Introduction to the Analytical List Page

Title Area

In the header area you can view information related to the Key Performance Indicator (KPI) or choose any of the following built-in SAP Fiori elements features to:

- Define or manage page variants
- Choose filter modes (compact or visual)
- Customize filter area
- Share analytical list page

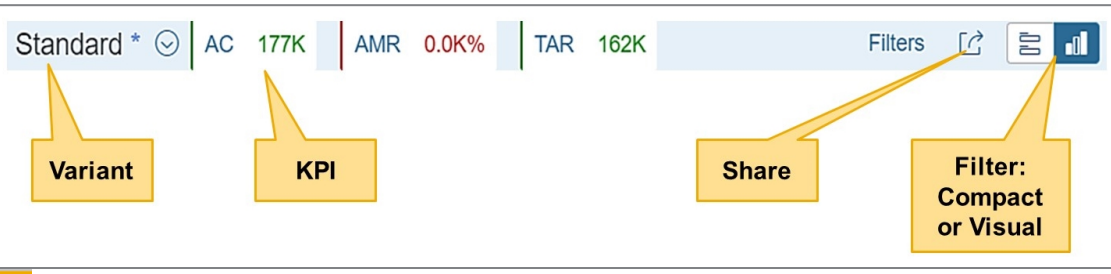


Figure 157: Title Area

Visual Filters

Visual Filter is an intuitive way of choosing filter values from an associated measure value. It supports line, bar, and donut chart types. Use visual filter to combine measures or item counts with filter values. Chart visualization increases the joy of use and the faster perception of relevant data. Selecting one or several chart data points allows quick analysis of the data set.

For example, to choose the country with the highest sales, you can visualize the graphic and make a selection on it. The default view of visual filter bar is based on the filter fields defined in the `SelectionFields` annotation for which a visual filter is defined.

© Copyright. All rights reserved.

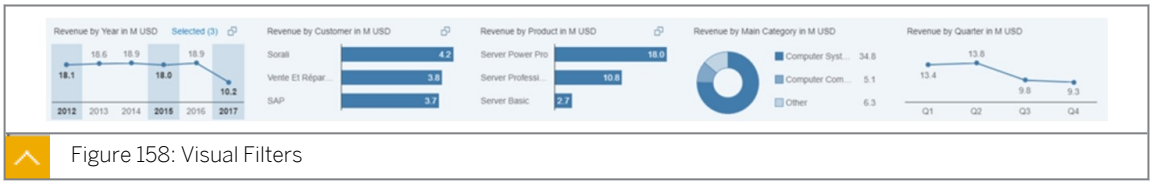


Figure 158: Visual Filters

Content Area

Users can interact with both the chart and the table. The initial view of the chart visualizes the most important aspects of the whole data set. Selecting a dimension within a chart area automatically filters all relevant information in the table area.

For example, if a chart selection is Country=ABC, then all records associated with this country selection are filtered in the table.

Note:
For detailed information for Analytic List Page Please access website: https://sapui5.netweaver.ondemand.com/sdk/#/topic and navigate to *Developing Apps with SAP Fiori Elements → Analytical List Page*.

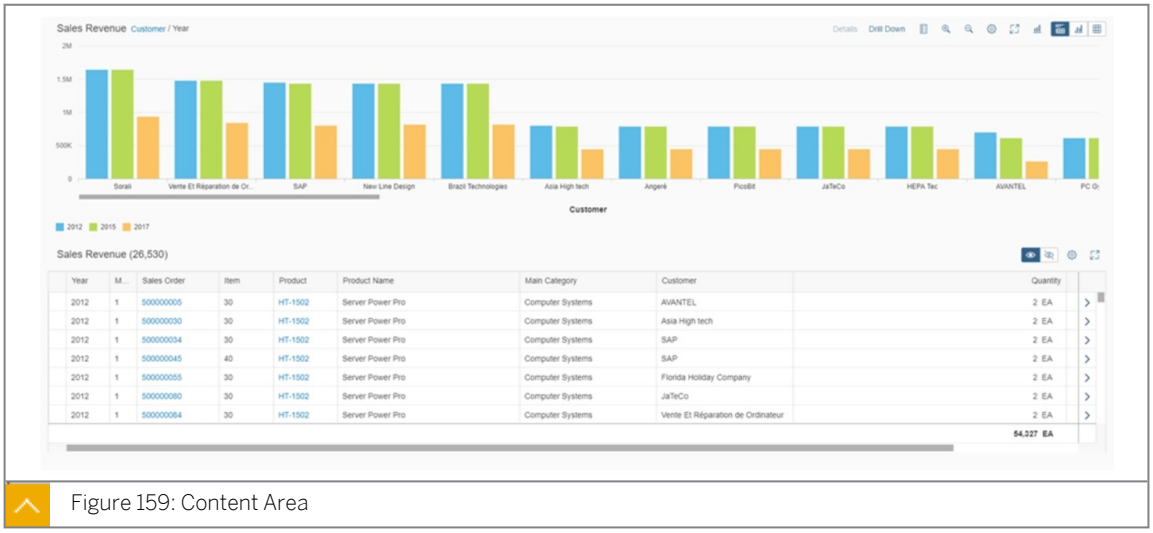


Figure 159: Content Area

ALP and Its Key Capabilities

Analytical Data

Data which is aggregate based, for example, sales data across different regions. Such aggregated data at different aggregation levels help in deeper analysis by allowing users to very easily focus on the areas with the highest (or lowest) values (for example, regions with the lowest sales values).

Analytical Data is designed by using **Analytical models**. Analytical models are recommended for reporting purposes when you need to use advantage of aggregations to expose results across different areas (for example, by time, by location, by responsible). These models are conceived over **Facts** and **Dimensions** and these views contain the basic data used to conduct detailed analysis and derive business values.

Imagine, as an example, a sales report which provides results based on customer, product, date, and sales person. The Fact is the sale itself and it holds values that we can measure (for example, number of sales and total amount of sales), the filters by

© Copyright. All rights reserved.

customer, product, time, and sales person are the Dimensions and these dimensions can have **Attributes** or **Texts** attached (for example, customer name, address, and product description) and when we connect all of them, we have a Cube and consequently an analytical model ready for consumption.

Transactional Data

Day-to-day records that could later be used for aggregation if required. For example, everyday sales records by which we can derive the sales region, the materials ordered, the suppliers, the amount of sales, and so on.

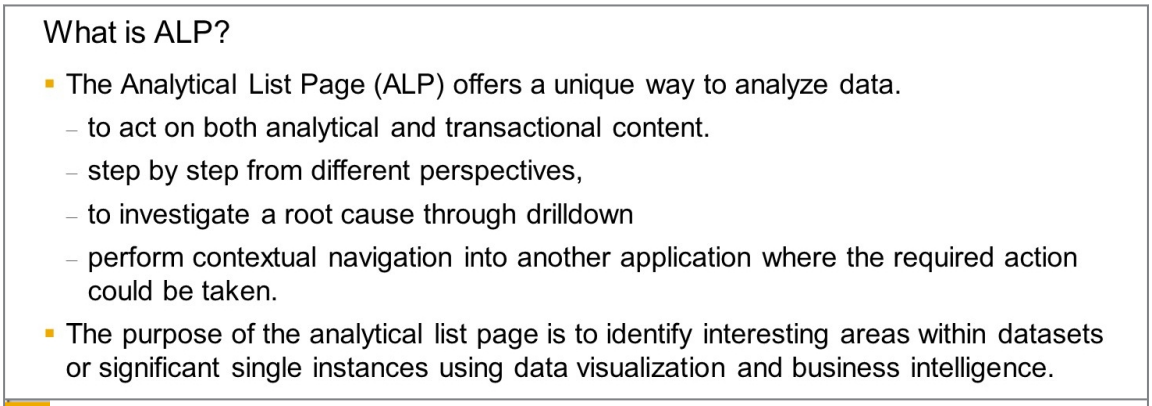


Figure 160: What is ALP?

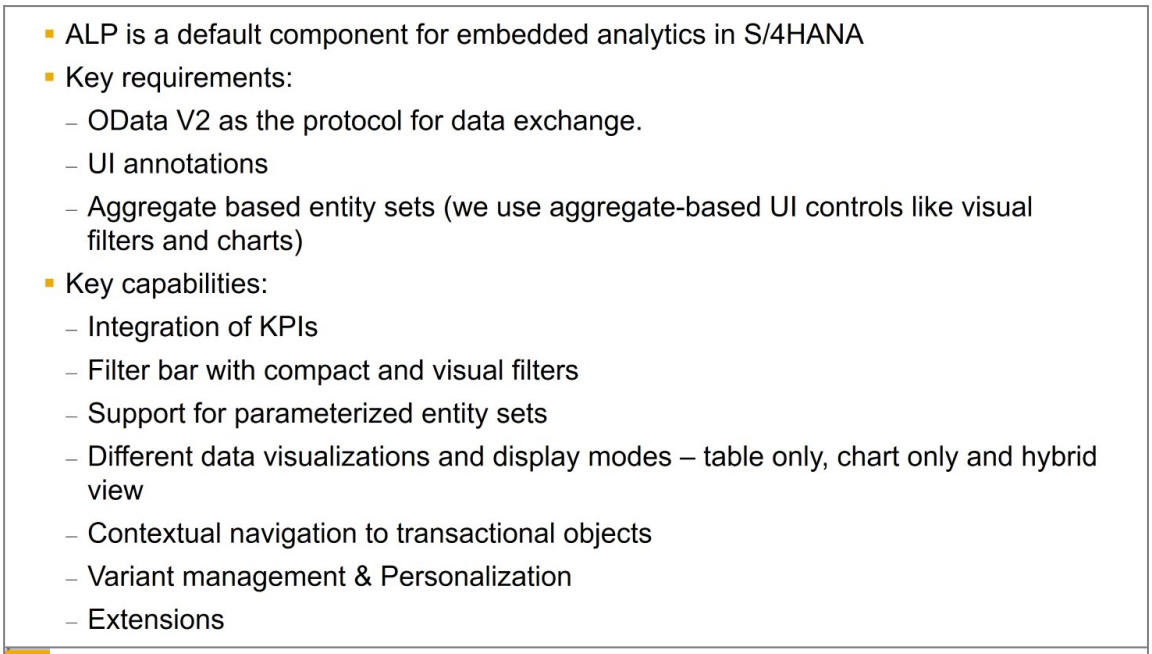


Figure 161: Key Characteristics

If applications want to bring in related information as KPIs, they can do one of the following:

- If the back end has a S/4HANA Smart Business, then the app-developer can only configure KPI tags using the set of KPIs exposed by the Smart Business KPI modeler (only these are shown when using the *Add KPI* plugin).
- If the back end is not S/4HANA Smart Business, then the app-developer can configure KPI tags using local annotations (having `UI.KPI` annotation) and making manual changes to the manifest. The *Add KPI* plugin cannot yet be used.

© Copyright. All rights reserved.

Guidance for the usage of the List Report and the Analytical List Page:

- List Report
  - Apps which allow CRUD operations on the object page (for example, by using Draft 2.0)
  - Master Data Lists (w/o any measure)
  - Apps which need multiple table views or multiple charts
  - Worklists (w/o Smart Filter Bar)
- Analytical List Page
  - Analytical Applications / Monitoring Applications / Complex content with multiple dimensions to be looked at
  - Apps which need KPIs
  - Apps which need visual filters
  - Apps which need a split view or an interaction option between chart and table

Figure 162: List Reports versus Analytic List Page

Implementing ALP

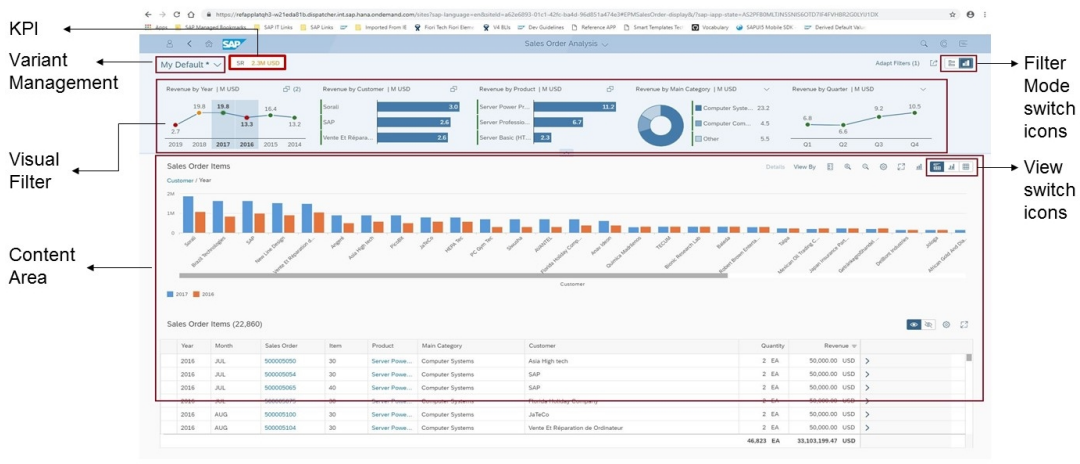


Figure 163: ALP and its Key Capabilities

As already described, ALP is based on a analytical model. So the basis for implementing an ALP is that you have implemented an analytical model.

As CDS query runs on Analytic Engine (in BW, it is called OLAP Engine), the data model should still align with traditional BW assumptions. CDS views just provide a faster, easier, more flexible way to build up such a data model.

The upper part of this picture shows the traditional objects in a typical BW system. To report on a BW system: InfoProviders are built by using Modeling Tool for BW on HANA, or RSA1 for BW on other DBs. Then, BW queries are built on these InfoProviders by using Query Designer, either Bex Query Designer or Query Designer in Modeling Tool.

It is similar for a transient query to run on the Analytic Engine in a non-BW system such as SAP S/4HANA. The lower part of the picture shows this. InfoProviders should be built by using CDS views, then CDS queries are built on top of it by using CDS views, but with different annotations.

© Copyright. All rights reserved.

Planning is not supported in customer defined CDS. See Note 2874534 - Planning on ABAP CDS View).

CDS views are just a new tool to build the data model for Analytic Engine. The only difference is, with traditional tools like RSA1, Query Designer or Modeling Tool, the boundary of different objects are well defined in the tools. The tools guide end users to build a valid model. However, CDS views is much lighter tool with strong capability, by reducing the Extraction, Transformation, and Loading (ETL) effort, a data model can be built very conveniently by typing some CDS scripts. But, the boundary of the objects are blurred. The explicit restrictions defined in traditional BW tools become implicit rules for users to follow in order to build a valid model. Although there is no terms like 'InfoProvider' or 'Query' in the CDS transient objects world, nevertheless users still need to bear this picture in mind while modeling a business scenario. With the concept, you will understand where to define what.

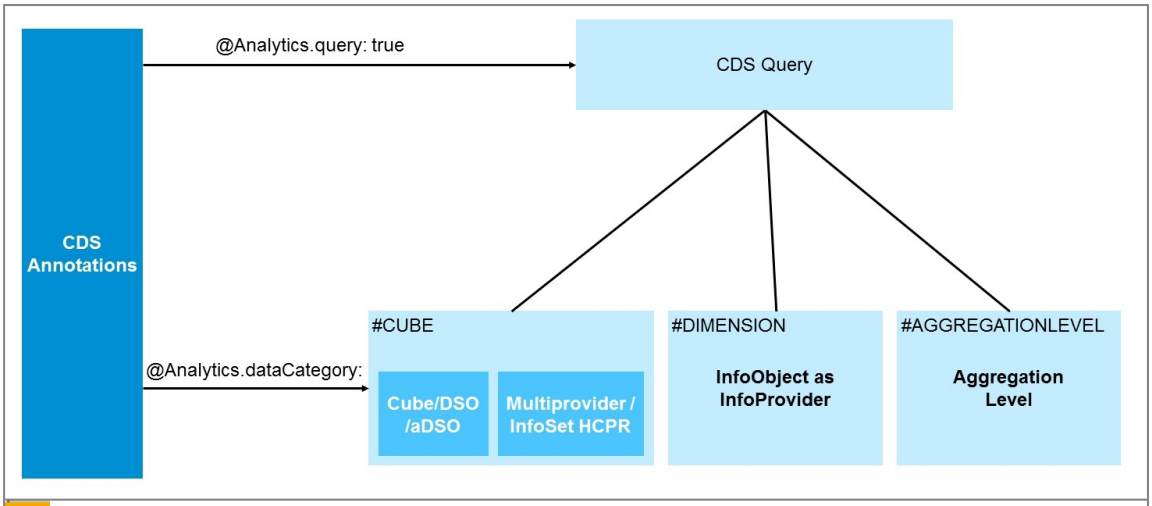


Figure 164: Understand Analytics on ABAP CDS View Data Models

After your analytical model is implemented on the SAP S/4HANA system, the following steps are needed to create a ALP using the SAP BAS:

1. Select *SAP Fiori Elements* as project template.
2. Choose *Analytical List Page* as Floorplan.
3. Select your data source and choose the OData-Service (ABAP CDS marked with `@OData.publish:true` and `@Analytics.query: true`).
4. Select the Main entity of your OData-Service and add the appropriate configuration.
5. Insert the general project settings.
6. After the project is generated, adapt the manifest.json configuration if necessary.

© Copyright. All rights reserved.

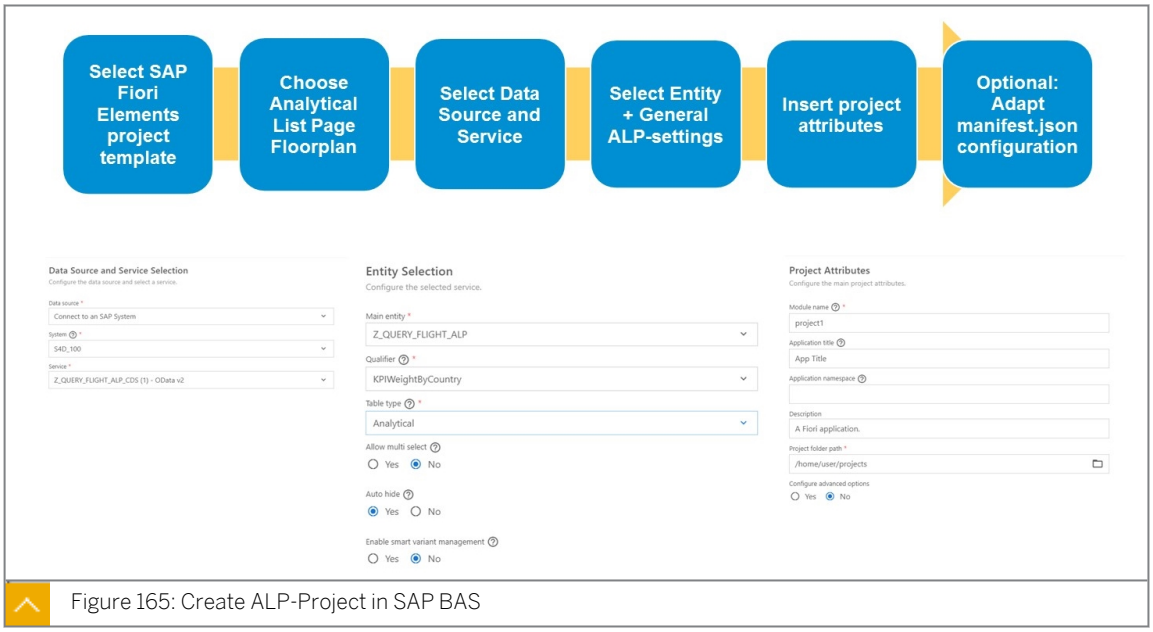


Figure 165: Create ALP-Project in SAP BAS

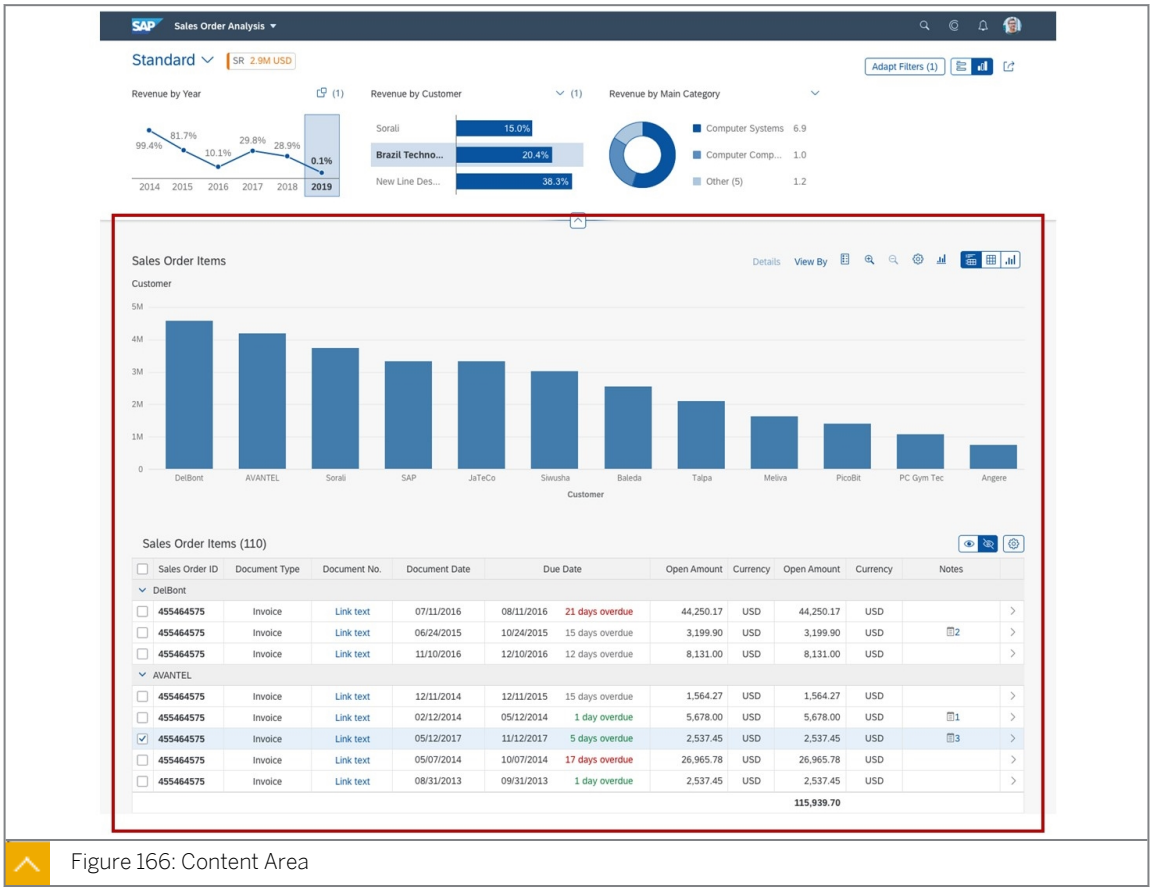


Figure 166: Content Area

Check https://sapui5.hana.ondemand.com/#/topic/2a9df06673d34f72b238549d49da8bfb for full list of manifest entries and what are the different values that they can take.

© Copyright. All rights reserved.

- Template level settings that can be influenced when creating an ALP based application:
  - `multiselect` (`default=false`)
  - `tableType` (`default=`*<ALP internal logic>*)
  - `qualifier` = <points to the SelectionPresentationVariant or a PresentationVariant annotation qualifier that influences the table and chart definition>
  - `smartVariantManagement` (`default=true`)
  - `showGoButtonOnFilterBar` (`default=false`)

```
"sap.ui.generic.app": {
    "_version": "1.3.0",
    "settings": {
        "forceGlobalRefresh": false,
        "objectPageHeaderType": "Dynamic",
        "showDraftToggle": false
    },
    "pages": {
        "AnalyticalListPage|ZCDS_UX403_ALPT_01": {
            "entitySet": "ZCDS_UX403_ALPT_01",
            "component": {
                "name": "sap.suite.ui.generic.template.AnalyticalListPage",
                "list": true,
                "settings": {
                    "condensedTableLayout": true,
                    "showGoButtonOnFilterBar": true,
                    "tableType": "AnalyticalTable",
                    "multiSelect": false,
                    "qualifier": "Default",
                    "autoHide": true,
                    "smartVariantManagement": false,
                    "keyPerformanceIndicators": {}
                }
            },
            "pages": {
                "ObjectPage|ZCDS_UX403_ALPT_01": {
                    "entitySet": "ZCDS_UX403_ALPT_01",
                    "defaultLayoutTypeIfExternalNavigation": "MidColumnFullScreen",
                    "component": {
                        "name": "sap.suite.ui.generic.template.ObjectPage"
                    }
                }
            }
        }
    }
},
```

Figure 167: Manifest Settings

Implement Visual Filters

A visual filter is an intuitive way of choosing filter values from an associated measure value. It supports line, bar, and donut chart types. Visual filter fields are always a subset of compact filters. This is because the compact filters dialog lists all the fields in the entity sets which are marked with `"sap:filterable"=true`. For this field to be rendered as a visual filter, we need in addition to this, the visual filter annotations (a valuelist annotation enhanced with `PresentationVariantQualifier`).

Use a visual filter to combine measures or item counts with filter values. Chart visualization increases the joy of use and the faster perception of relevant data. Selecting one or several chart data points allows quick analysis of the data set.

For example, to choose the country with the highest sales, you can visualize the graphic and make a selection on it. The default view of a visual filter bar is based on the filter fields defined in the `SelectionFields` annotation for which a visual filter is defined.

© Copyright. All rights reserved.

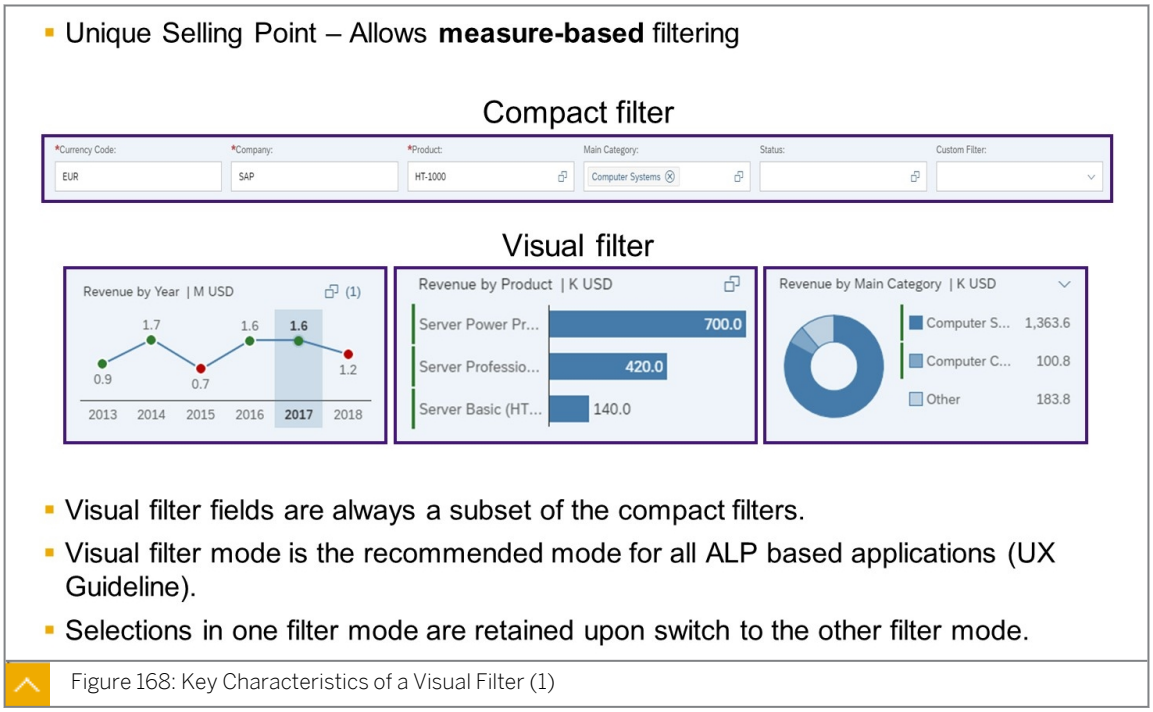


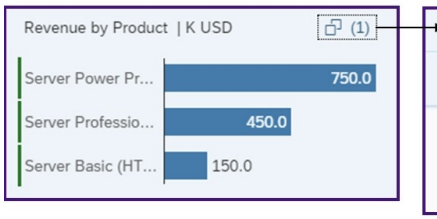
Figure 168: Key Characteristics of a Visual Filter (1)

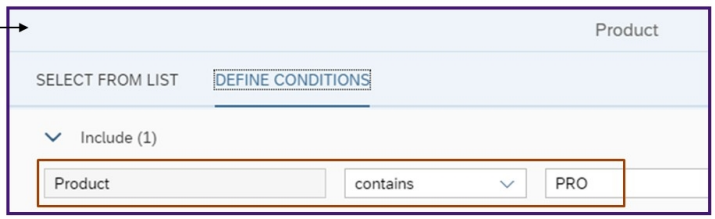
There is no support for using compact filters to reduce data set for visual filters for the SAME filter field. For example, there is *Product* as a filter field in both the compact filter as well as visual filters (say a bar chart *Revenue by Product* showing top three records). Let's say, the user now tries to add a complex condition filter in the compact filter for Product: (Product contains *PRO*) and now switches to the visual filter.

The user might expect that only the products which contains *PRO* within it are considered for bringing up the visual filter but this is not the case, and it is so by design. The visual filter is meant to show the top three products irrespective of the value for the same field in the compact filter. Nevertheless, the complex condition is seen when you select the (1) in the top right of the visual filter and it is correctly carried forward to the content area.

In the example shown in the figure, the first two records have *PRO* within their description and the user might expect that they are shown as highlighted, but at the moment this does not happen. Again, the complex condition that was applied is correctly carried forward to the content area. However, currently, we do not yet highlight the chart selections based on a complex condition coming via the compact filters.

© Copyright. All rights reserved.





- Known restrictions:
  - Complex conditions: Seen only in top right area of the visual filters. Conditions carried forward to content area but do not show up as selected for matching records seen in visual filter chart.
  - “Other” section in donut cannot be selected together with other sections of the donut.
  - “Other” section in donut cannot be selected in case the field is dropdown based.
  - Interval based fields (`sap:filter-restriction="interval"`) are ignored.

Figure 169: Key Characteristics of a Visual Filter (2)

Visual Filter supports semantic coloring. Both `UI.Criticality` and `UI.CriticalityCalculation` annotations are supported. The first one allows applications to have their own back end logic to calculate the semantic color for a given data point value.

The second one allows applications to specify either fixed or dynamic values that the ALP framework would use to apply semantic color on the chart data points (https://sapui5.hana.ondemand.com/#/topic/1714720cae984ad8b9d9111937e7cd38).

Applications cannot influence the scale factor, ALP determines the best scale factor via SAPUI5 APIs based on the values plotted on the chart.

The ALP ignores the *UI.Hidden* fields when you select filters if the IN mapping points to a field marked with *UI.Hidden* in the valuelist entity set. For example, the *Status_ID* from the main entity set points to *StatusCode* in the value help entity set (of the visual filter). If the *StatusCode* is marked as *UI.Hidden*, then the incoming value is ignored (https://sapui5.hana.ondemand.com/#/topic/16d43eb0472c4d5a9439ca1bf92c915d).

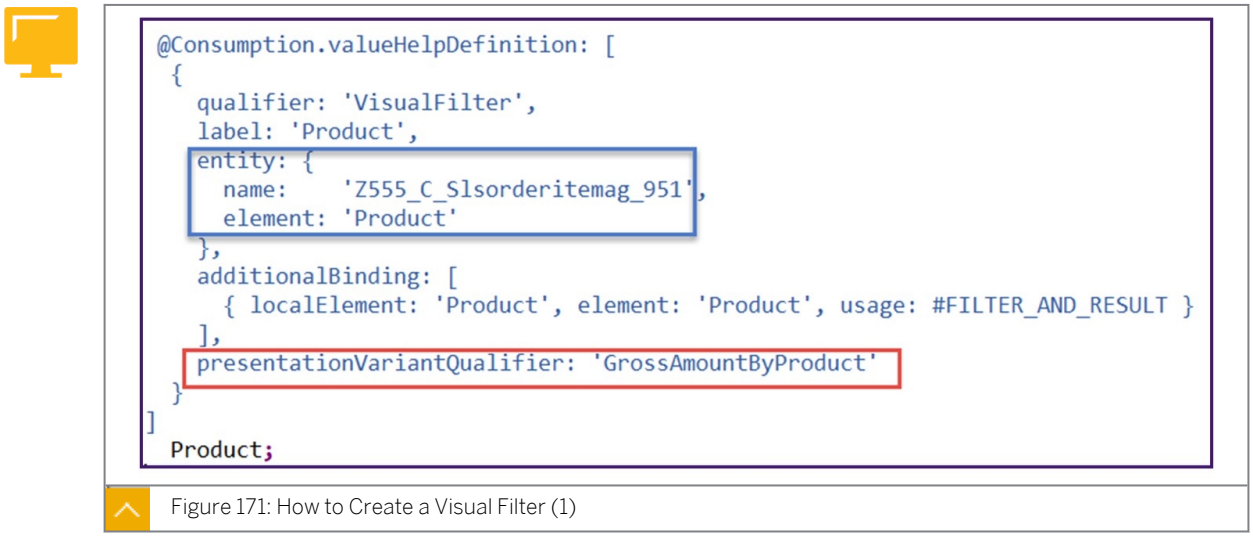
- Pre-determined sorting criteria – CANNOT be changed by design!
- Support for semantic coloring.
- Auto determination of scale factor.
- Decimal precision – 0 or 1 ONLY. Applications can influence via UI.DataPoint.NumberOfFractionalDigits connected to the visual filter.
- Adapt Filter Dialog & Personalization.
- Selection in one visual filter could be used as a filter to refresh another visual filter.
  It is enabled via IN mapping within the Common.Valuelist annotation.
- Visual filters could be configured to show always a fixed set of values rather than the top 3 or bottom 3 values.

Figure 170: Key Characteristics of a Visual Filter (3)

© Copyright. All rights reserved.

The following steps show you how to create a visual filer.

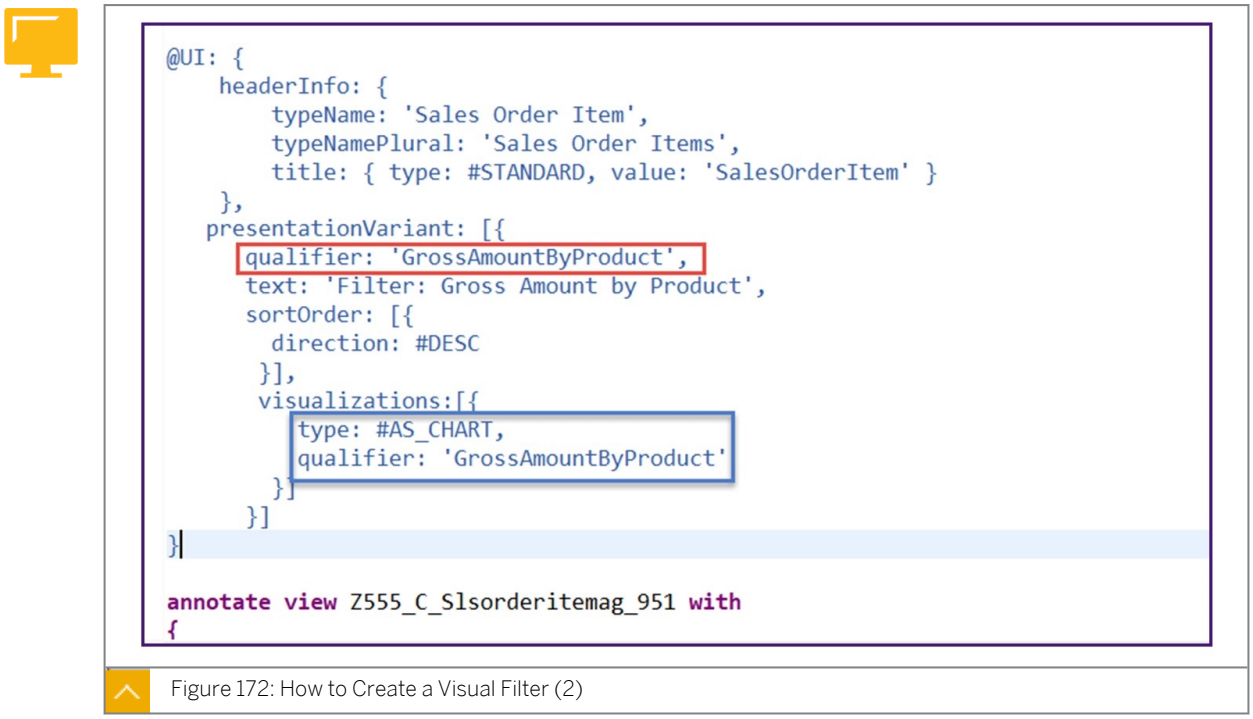
Add `@Consumption.valueHelpDefinition` annotation to the property that should become a visual filter.



```
@Consumption.valueHelpDefinition: [
 {
   qualifier: 'VisualFilter',
   label: 'Product',
   entity: {
     name:    'Z555_C_Slsorderitemag_951',
     element: 'Product'
   },
   additionalBinding: [
     { localElement: 'Product', element: 'Product', usage: #FILTER_AND_RESULT }
   ],
   presentationVariantQualifier: 'GrossAmountByProduct'
 }
]
 Product;
```

Figure 171: How to Create a Visual Filter (1)

Add `@UI.presentationVariant` annotation to the view.



```
@UI: {
    headerInfo: {
        typeName: 'Sales Order Item',
        typeNamePlural: 'Sales Order Items',
        title: { type: #STANDARD, value: 'SalesOrderItem' }
    },
   presentationVariant: [{
      qualifier: 'GrossAmountByProduct',
      text: 'Filter: Gross Amount by Product',
      sortOrder: [{
        direction: #DESC
       }],
       visualizations:[{
          type: #AS_CHART,
          qualifier: 'GrossAmountByProduct'
        }]
       }]
}

annotate view Z555_C_Slsorderitemag_951 with
{
```

Figure 172: How to Create a Visual Filter (2)

Add the `@UI.chart` annotation.

© Copyright. All rights reserved.

```
@UI: {
    headerInfo: {
        typeName: 'Sales Order Item',
        typeNamePlural: 'Sales Order Items',
        title: { type: #STANDARD, value: 'SalesOrderItem' }
    },
    chart:[
      {
      qualifier: 'GrossAmountByProduct',
      chartType: #BAR,
      title: 'GrossAmount by Product',
      description: 'GrossAmount by Product',
      dimensions:  [ 'Product' ],
      dimensionAttributes: [ { dimension: 'Product', role: #CATEGORY } ],
      measures:  [ 'GrossAmount' ],
      measureAttributes: [ { measure: 'GrossAmount', role: #AXIS_1 }  ]
      }
  ],
   presentationVariant: [{
      qualifier: 'GrossAmountByProduct',
      text: 'Filter: Gross Amount by Product',
      sortOrder: [{
        direction: #DESC
       }],
       visualizations:[{
          type: #AS_CHART,
          qualifier: 'GrossAmountByProduct'
        }]
      }]
}

annotate view Z555_C_Slsorderitemag_951 with
{
```

Figure 173: How to Create a Visual Filter (3)

CDS annotation `@Consumption.valueHelpDefinition` will be transferred to the `OData Common.ValueList` annotation.

```
<Annotations xmlns="http://docs.oasis-open.org/odata/ns/edm" Target="Z555_C_SLSORDERITEMAG_951_CDS.Z555_C_Slsorderitemag_951Type/Product">
  <Annotation Term="Common.ValueList" Qualifier="VisualFilter">
    <Record>
      <PropertyValue Property="Label" String="Product"/>
      <PropertyValue Property="CollectionPath" String="Z555_C_Slsorderitemag_951"/>
      <PropertyValue Property="SearchSupported" Bool="true"/>
      <PropertyValue Property="PresentationVariantQualifier" String="GrossAmountByProduct"/>
      <PropertyValue Property="Parameters">…</PropertyValue>
    </Record>
  </Annotation>
  <Annotation Term="Common.FieldControl" EnumMember="Common.FieldControlType/Mandatory"/>
</Annotations>
```

Figure 174: How to Create a Visual Filter (4)

- Filters shown in the filter bar by default: `UI.SelectionFields` annotation
- Visual Filter definition: `Common.valuelist` annotation

Business Object	Z0020 (Main Entity Set)	Z0021 (Entity Set)	Z0022 (Entity Set)	Z0023 (Entity Set)
Region	RegionID	RegionCode	RegionIdentifier	RegionNumber
Country	CountryID	CountryCode	CountryIdentifier	CountryNumber
Plant	PlantID	PlantCode	PlantIdentifier	PlantNumber

Annotation Example

```
<Annotations Target="Z0020_CDS.Z0020Type/RegionID"> // specifies the ValueList annotation is on the "RegionID" dimension in Z0020 entity set
<Annotation Term="Common.ValueList" Qualifier="0020_Region">
<Record>
            <PropertyValue Property="Label" String="Region"/>
            <PropertyValue Property="CollectionPath" String="Z0021"/> // Specifies that the entity set for fetching the value help is Z0021
            <PropertyValue Bool="false" Property="SearchSupported"/>
            <PropertyValue Property="Parameters"> // In and Out parameters to map the main entity set with the value help entity set
            <Collection>
                  <Record Type="Common.ValueListParameterOut"> // OUT  parameter determine elements in the main entity set that make it to the filter query and also determine the entity from the value
                        <PropertyValue Property="LocalDataProperty" PropertyPath="RegionID"/> // LocalDataProperty points to an entity in the main entity set - this is the entity used in forming the
                        <PropertyValue Property="ValueListProperty" String="RegionCode"/> // ValueListProperty refers to the entity within the value help entity set that provides the "value" for the
                  </Record>
                  <Record Type="Common.ValueListParameterIn"> // IN parameters help in filtering the data set of the value help entity set
                        <PropertyValue Property="LocalDataProperty" PropertyPath="PlantID"/>
                        <PropertyValue Property="ValueListProperty" String="PlantCode"/>
                  </Record>
            </Collection>
            </PropertyValue>
            <PropertyValue Property="PresentationVariantQualifier" String="Bar_0021_Region"/> // The presence of a presentation variant means Region comes up as a visual filter.
      </Record>
</Annotation>
</Annotations>
```

Figure 175: Annotation (1)

© Copyright. All rights reserved.

- Semantic coloring

Figure 176: Annotation (2)

- The `UI.Chart` annotation must define in the `MeasureAttributes` section a reference to a `DataPoint`. The `DataPoint` will contain then the criticality definition.

```
<PropertyValue Property="MeasureAttributes">
    <Collection>
        <Record>
            <PropertyValue Property="Measure" PropertyPath="RecognizedMargAmountInDisplayCurrency"/>
            <PropertyValue Property="DataPoint" AnnotationPath="@UI.DataPoint#Calculated"
        </Record>
    </Collection>
</PropertyValue>
```

```
<Annotation Term="UI.DataPoint" Qualifier="Calculated">
    <Record Type="UI.DataPoint">
        <PropertyValue Property="Value" Path="Course"/>
        <PropertyValue Property="Criticality" Path="CourseCriticality"/>
    </Record>
</Annotation>
```

Figure 177: Annotation (3)

- Data point with Criticality Calculation

```
<Annotation Term="UI.DataPoint" Qualifier="DP_2">
    <Record Type="UI.DataPointType">
        <PropertyValue Property="Value" Path="RecognizedMargAmtInDisplayCrcy"/>
        <PropertyValue Property="CriticalityCalculation">
            <Record Type="UI.CriticalityCalculationType">
                <PropertyValue Property="ImprovementDirection"
                    EnumMember="UI.CriticalityCalculationType/Maximize"/>
                <PropertyValue Property="AcceptanceRangeLowValue" Decimanl="1000000"/>
                <PropertyValue Property="ToleranceRangeLowValue" Int="5000"/>
                <PropertyValue Property="DeviationRangeLowValue" Int="1000"/>
            </Record>
        </PropertyValue>
    </Record>
</Annotation>
```

Figure 178: Annotation (4)

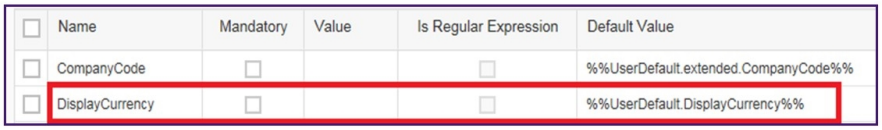
© Copyright. All rights reserved.

- An overlay will be shown over each graphical filter which is based on amounts as measures and where no unique currency value is available

- Therefore, the central guidance is the following:
  - Make use of the User Default Parameter “Display Currency” (technical fieldname `DisplayCurrency`) in your app.

Figure 179: Currency Handling (1)

- The FLP content expert should maintain a target mapping for the parameter `DisplayCurrency`.



Name	Mandatory	Value	Is Regular Expression	Default Value
CompanyCode				%%UserDefault.extended.CompanyCode%%
DisplayCurrency				%%UserDefault.DisplayCurrency%%

- Offer ‘Display Currency’ as a visible field in the Smart Filter Bar.

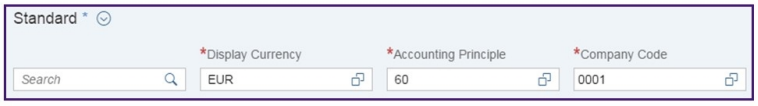


Figure 180: Currency Handling (2)

- The field “Display Currency” should be a parameter in your CDS view and it should be used for triggering a currency conversion into a measure “Amount in Display Currency” (adjust the description for your app).

```
define view Z2_Trrmonitor
  with parameters P_DisplayCurrency : vdm_v_display_currency

  as select from I_Trrmonitor

{
          ....
          @Semantics.currencyCode:true
          @UI.hidden: true
          cast($parameters.P_DisplayCurrency as vdm_v_display_currency ) as DisplayCurrency,

          @DefaultAggregation: #SUM
          @Semantics.amount.currencyCode: 'DisplayCurrency'
          cast( cast(currency_conversion (
             amount => RecognizedMarginAmtInCCCrcy,
             source_currency => CompanyCodeCurrency,
             target_currency => $parameters.P_DisplayCurrency,
             exchange_rate_date => cast($session.system_date as abap.dats ) )
             as abap.curr(23,2) ) as farp_amount_display_crcy ) as AmountInDisplayCurrency,
```

- The visual filters should use as the measure “Amount in Display Currency”.
- Optionally offer also the column “Amount in Display Currency” in the Table / Chart in the main content area.

Figure 181: Currency Handling (3)

The following figure shows an example definition for a value help for a visual filter.

© Copyright. All rights reserved.

```
        <PropertyValue Property="CollectionPath" String="Z2_TrrmonitorResults"/>
        <PropertyValue Property="SearchSupported" Bool="false"/>
        <PropertyValue Property="Parameters">
            <Collection>
                <Record Type="com.sap.vocabularies.Common.v1.ValueListParameterInOut">
                    <PropertyValue Property="ValueListProperty" String="Project"/>
                    <PropertyValue Property="LocalDataProperty" PropertyPath="Project"/>
                </Record>
                <Record Type="com.sap.vocabularies.Common.v1.ValueListParameterDisplayOnly">
                    <PropertyValue Property="ValueListProperty" String="AmountInDisplayCurrency"/>
                    <PropertyValue Property="LocalDataProperty" String="AmountInDisplayCurrency"/>
                </Record>
                <Record Type="com.sap.vocabularies.Common.v1.ValueListParameterIn">
                    <PropertyValue Property="ValueListProperty" String="AccountingPrinciple"/>
                    <PropertyValue Property="LocalDataProperty" PropertyPath="AccountingPrinciple"/>
                </Record>
                <Record Type="com.sap.vocabularies.Common.v1.ValueListParameterIn">
                    <PropertyValue Property="ValueListProperty" String="CompanyCode"/>
                    <PropertyValue Property="LocalDataProperty" PropertyPath="CompanyCode"/>
                </Record>
            </Collection>
        </PropertyValue>
        <PropertyValue Property="PresentationVariantQualifier" String="VH_2"/>
```

Figure 182: Filter Influencing Another

FLP default values are applied only in non-navigation context, for example, if you launch an ALP app directly from an FLP tile, there is no xapp-state navigation involved.

- Filter bar defaults can come via multiple sources
  - CDS/OData metadata defaults (`Common.FilterDefaultValue`)
  - Annotation: `UI.SelectionVariant`
  - User default variant
  - Incoming navigation context
  - FLP defaults
- FLP default values are applied only in non-navigation context (i.e. when ALP is launched via static tile)
  - Overrides fully values from CDS/OData metadata and SV annotation only in case of no default variant.
- In navigation context, only `DisplayCurrency` from FLP settings is used (if there is nothing coming in navigation context)

Figure 183: Default Values and Navigation

Implement Title Area

In the header area, you can view information related to the Key Performance Indicator (KPI) or choose built-in SAP Fiori element features to carry out the following tasks:

- Define or manage page variants
- Choose filter modes (compact or visual)
- Customize filter area
- Share analytical list page

© Copyright. All rights reserved.

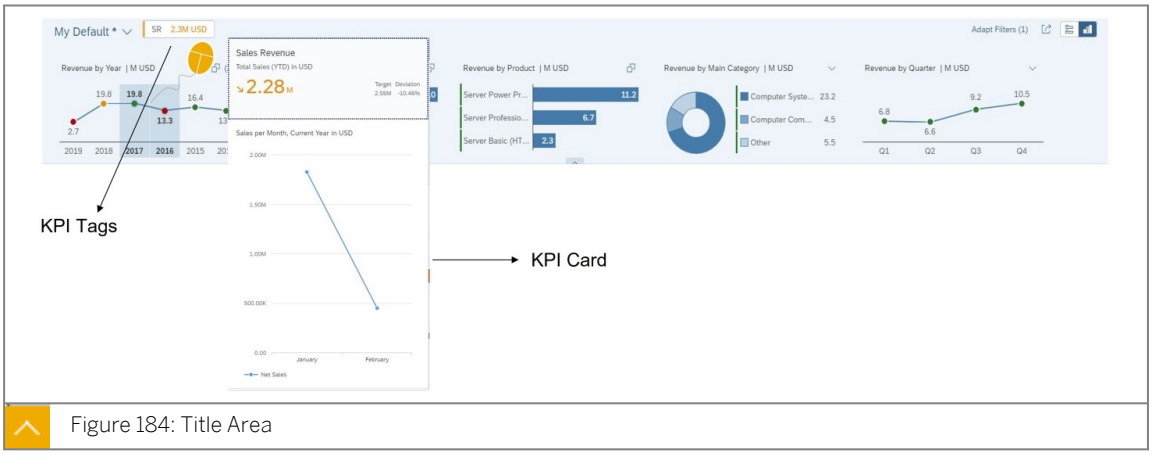


Figure 184: Title Area

KPIs (Key Performance Indicators) allow applications to bring in related key indicators (Outstanding Orders, Operating Margin, Net Profit, and so on). This enables the end user to get a quick picture on the overall state of the system, especially given the fact that the KPIs can be semantically colored to reflect the health of the KPI.

Global KPIs do not react to changes in the filter bar whereas the Filterable/Local KPIs will react to filter bar changes. For example, a selection of a particular year in the filter bar will lead to NetSales filterable KPI showing only the net sales for the chosen year (assuming that the filter bar *Year* field is also present in the KPI entity set as a filterable field of course).

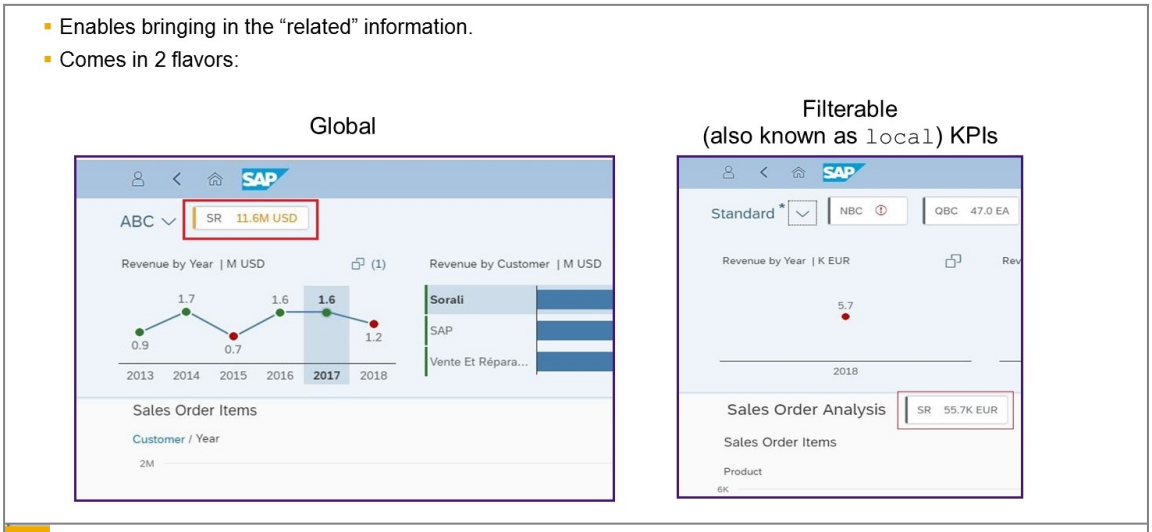


Figure 185: Key Characteristics of KPI Tags (1)

Note that if there is hard-coded criticality defined for the KPI, then such KPIs would not be good candidates for the filterable KPIs (since the semantic coloring against hard-coded threshold value cannot be applied when the KPI value itself is being changed due to filters being applied). For example, it makes no sense to have a fixed `SalesRevenueTarget` if the SalesRevenue KPI itself is being changed due to filters being applied. It only makes sense to semantically color the *SalesRevenue* KPI if the *SalesRevenueTarget* is also changing accordingly when the filters are applied (that is, threshold values are not static).

© Copyright. All rights reserved.

- Supports semantic coloring
- Progressive Disclosure. KPI Tags → KPI Cards → KPI Drilldown (Typically SAP Smart Business,SSB, drilldown page)

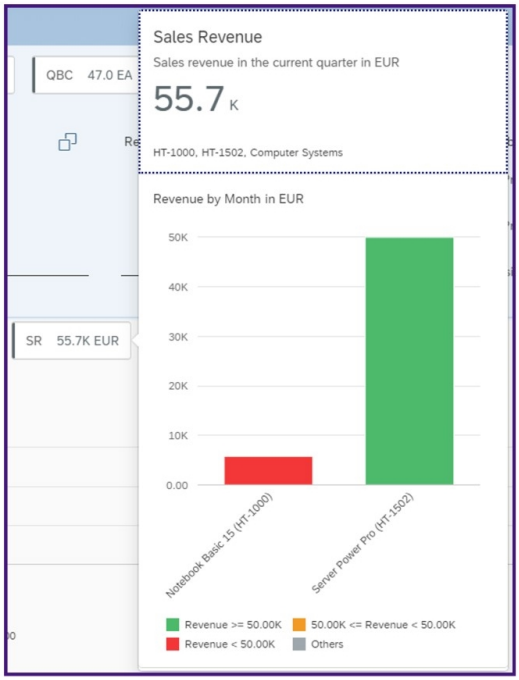


Figure 186: Key Characteristics of KPI Tags (2)

Figure 187: Annotation for KPIs

© Copyright. All rights reserved.

- KPIs are normally created via SSB tool (“Create KPI” SSB modeler app).
- SSB (only S/4 Hana version) creates the UI.KPI annotation that defines each KPI.
- ALP should allow only SSB defined KPIs!
  - This is why `UI.KPI` annotation is not enabled from CDS.
  - For non SAP S/4HANA scenarios, use local annotation and manual steps.

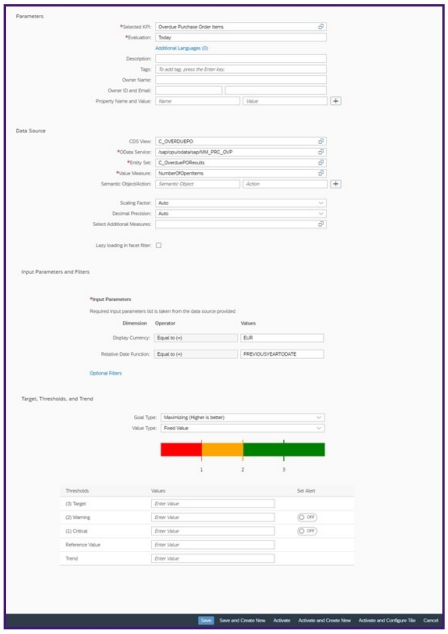


Figure 188: Integration Between ALP and SSB (1)

- Web IDE “Add KPI” plugin checks if backend has SSB persistency (true only if SAP S/4HANA SSB is installed).
- If present, we list all available KPIs in the system.
  - This discovery mechanism is limited ONLY to SAP S/4HANA SSB!
  - The other flavors of SSB do not yet have the development required for the integration.
- For systems that do not have SAP S/4HANA SSB installed, local annotations and manual steps need to be used.

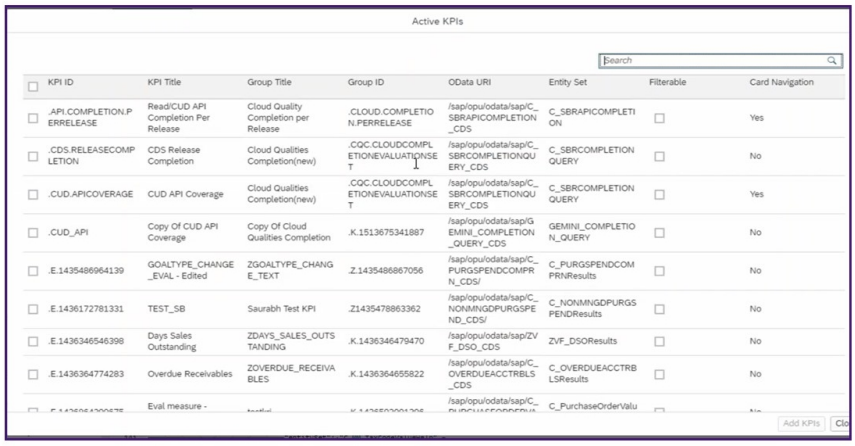


Figure 189: Integration Between ALP and SSB (2)

© Copyright. All rights reserved.

Advanced Topics

- Navigation to custom actions defined via manifest.
- Navigation to target defined via annotation `DataFieldForIntentBasedNavigation`
- Navigation to target defined via Smart Links. Via `SemanticObject` annotation against the field used in the table / against the data point selected in the chart.
- Navigation to details from table
  - By default, this is the object page navigation (has to be same entity set as main entity set of ALP!).
  - Can be changed to an external app navigation (SO/Action based), exactly the same way as it is in LR.
  - Target can be changed for each record via `onListNavigationExtension` extension.
- Navigation possibility directly into object page (deep linking).

Figure 190: Navigation from ALP

Some applications have performance concerns loading the visual filter by default. They instead want to load the compact filters by default (needs exceptional UX approval!). However, even when loading the application with compact filter as default filter mode, the visual filter calls are normally made in the background (in order to have a faster E2E time when user switches to the visual filter mode). If applications choose to not do such visual filter calls in the background until when user really wants to switch to visual filter mode, they can then lazy load the visual filters.

- ALP is complex in nature – it deals with many aggregate based controls (visual filters, charts, table aggregates) which need full table scan in backend.
- One can lazy load the visual filters (view Visual Filter Setup) for faster initial response time if compact filters are the default filter mode and lower the load on the backend
  - But this means lower E2E response time when user switches to visual filters.
  - UX Guidelines recommends the visual filters as the default filter mode.
- On load of ALP – 3 batch calls (1 for visual filters, 1 for table and 1 for chart).
  - If KPIs are present, there will be further calls.
  - Currently there are separate batch calls for the chart and table and there are plans to optimize this in future.

Figure 191: Performance

LESSON SUMMARY

You should now be able to:

- Get an overview of the Analytical List Page

© Copyright. All rights reserved.

© Copyright. All rights reserved.

Learning Assessment

1. Which areas are part of an Analytic List Page?

   Choose the correct answers.

   - [ ] **A** Title Area
   - [ ] **B** Visual Filters
   - [ ] **C** Selection Fields
   - [ ] **D** Content Area

2. In an Analytic List Page, the user can filter data by selecting some points in the chart.

   Determine whether this statement is true or false.

   - [ ] True
   - [ ] False

© Copyright. All rights reserved.

Learning Assessment - Answers

1. Which areas are part of an Analytic List Page?

   Choose the correct answers.

   - [X] A Title Area
   - [X] B Visual Filters
   - [ ] C Selection Fields
   - [X] D Content Area

   Correct. Title Area, Visual Filters, and Content Area are part of an Analytic List Page.

2. In an Analytic List Page, the user can filter data by selecting some points in the chart.

   Determine whether this statement is true or false.

   - [X] True
   - [ ] False

   Correct. In an Analytic List Page, the user can filter data by selecting some points in the chart.

© Copyright. All rights reserved.

UNIT 8 SAP Fiori Elements Applications

Lesson 1

UNIT OBJECTIVES

- Extending and Adapting SAP Fiori Elements Applications

© Copyright. All rights reserved.

© Copyright. All rights reserved.

Unit 8
Lesson 1

Extending and Adapting SAP Fiori Elements Applications

LESSON OBJECTIVES
After completing this lesson, you will be able to:

- Extending and Adapting SAP Fiori Elements Applications

Extend SAP Fiori Elements Based Apps

The following two options are provided to extend an app:

- App extension: Developers can add extensions during SAP Fiori elements-based apps using the Guided Development of SAP Business Application Studio and the extension API.
- Adaption extensions: Customers and Partners introduce their own functionality to an existing app using Adaption project of SAP Business Application Studio to build their own upgrade-safe application variant.

Generated SAP Fiori Element applications can be implemented using the provided SAPUI5-extensiblity features. Use app extensions with caution and only if you cannot implement the requirement using annotations or manifest settings. Adding an extension will move the responsibility to the application.

Go through https://ui5.sap.com/#/topic/d9c146a4e0f049108cf8231bfca5585b.html before using the extensibility features.

The SAP Fiori elements Guided Development features of SAP BAS are used to add visual and functional aspects to an existing SAP Fiori elements project.

© Copyright. All rights reserved.

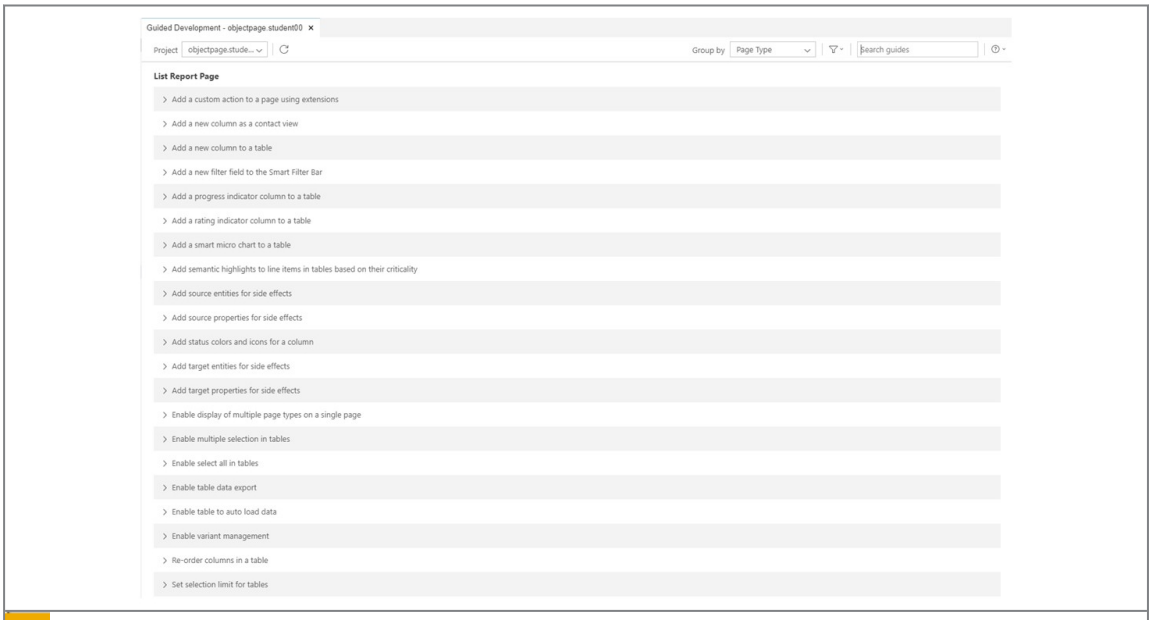


Figure 192: SAP Fiori Elements Guided Development

Extending List Report

The following list shows you the extension scenarios for list reports:

- Add custom fields to the Smart Filter Bar
- Add custom actions or popups
- Add additional columns to the Smart Table

Implementing Your Extensions

You can change your own control in the extension in any way you want. You can access it by its ID to change the text. Do not access or modify controls generated by SAP Fiori elements, use *extensionAPI* instead.

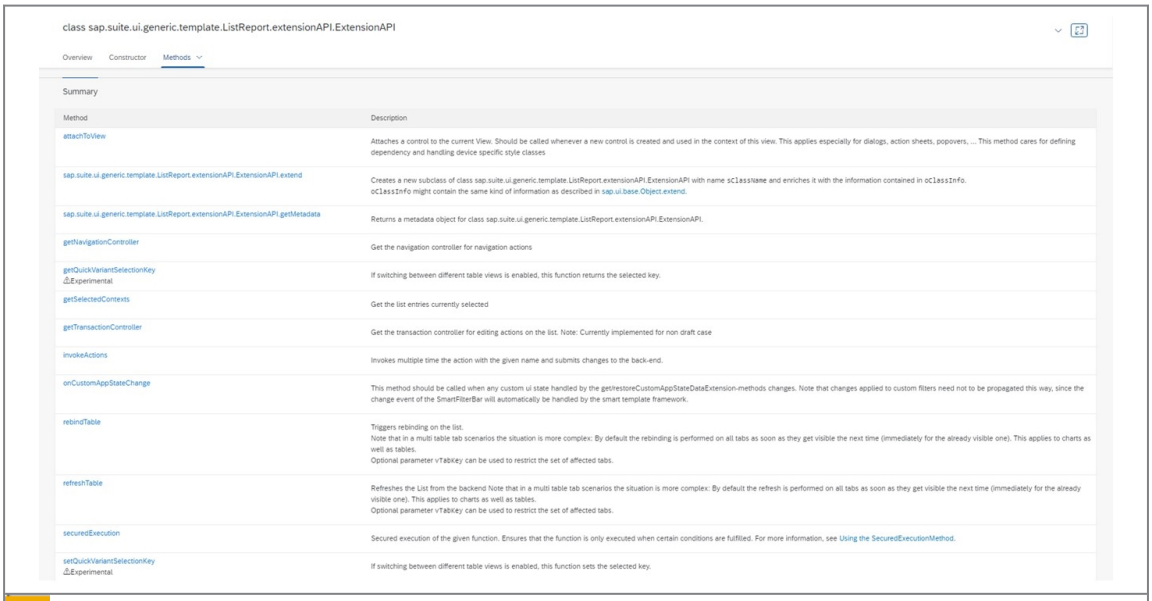


Figure 193: Implementing Your Extensions

© Copyright. All rights reserved.

Controls generated by FE shall not be accessed by the app coding. Even if the app just reads the property – the value can be changed by FE afterwards (so it cannot be reliable). The *extensionAPI* of the *ListReport-Template* provides various functions to extend the standard list report. For details, navigate to the following site: https://ui5.sap.com/#/api/sap.suite.ui.generic.template.ListReport.extensionAPI.ExtensionAPI

The guided development of SAP Business Application studio allows you to add custom actions to the list report.

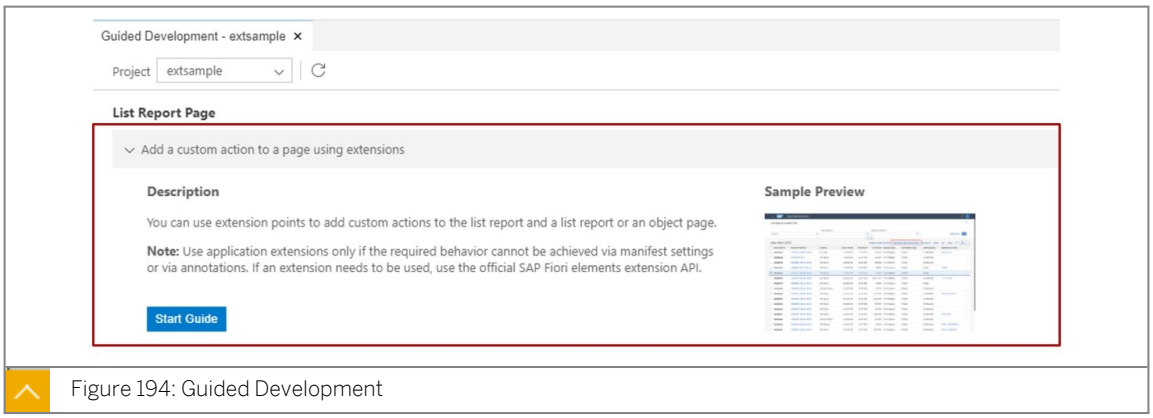


Figure 194: Guided Development

Adding Function Details

In the first step the developer selects the details of the new action. The page type dropdown allows the selection if the action should be added to the list report page or to the object page. The *Entity Set* selection allows you to choose to which entity set the action refers. The function name field allows to insert the name of the function.

The name of the function will be used as prefix for the function name the name of the selected entity will be used as a suffix. If no entity set is selected the suffix is `REPLACE_WITH_ENTITY_SET`.

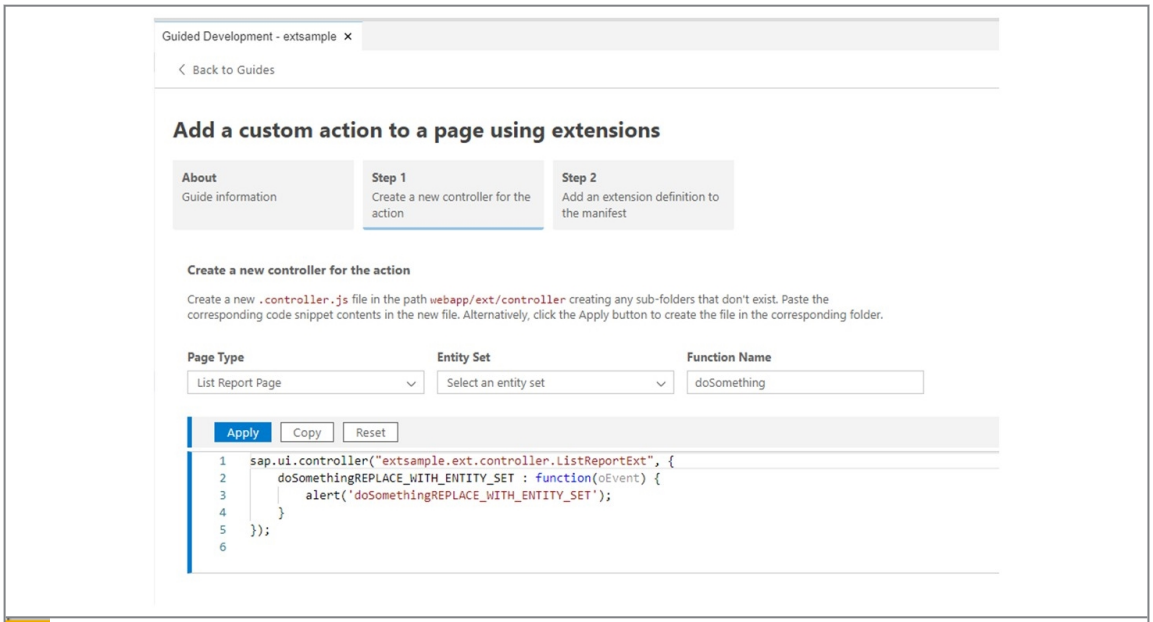


Figure 195: Add Function Details

In the next step the developer adds the configuration for the custom action.

© Copyright. All rights reserved.

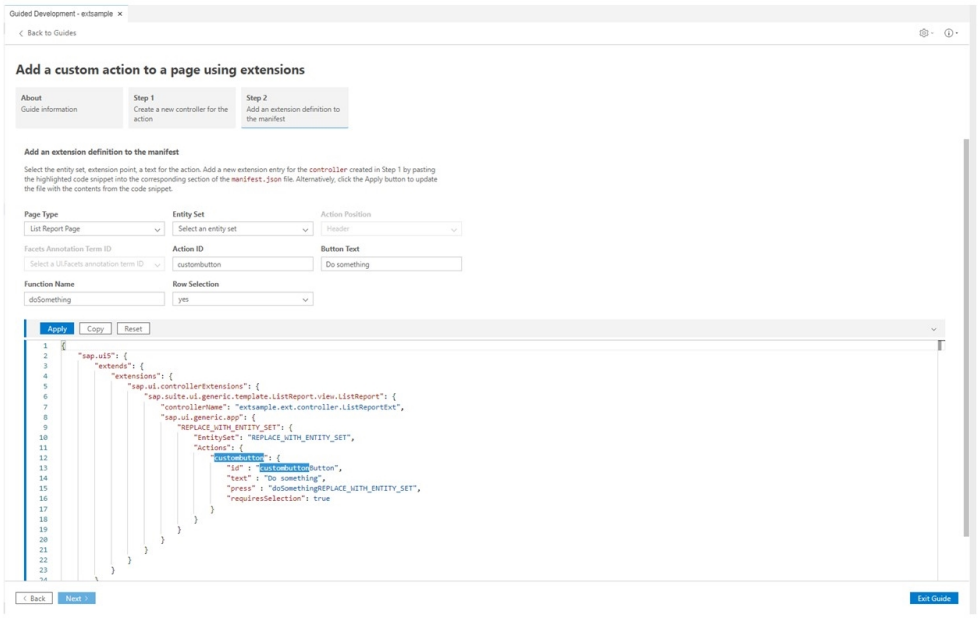


Figure 196: Action Configuration

The `manifest.json` contains the configuration for the extension of the list report.

Figure 197: manifest.json Configuration

The guided development added a new folder with the name controller under a new folder with the name ext. The controller implementation for the controller extension is added to the controller folder. Inside the controller you can now use the *extensionAPI-functions*.

© Copyright. All rights reserved.

Figure 198: Generated Controller Extension

securedExecution of extensionAPI

Use the `securedExecution` method whenever one of the following operations must be performed by extension coding:

- An asynchronous operation
- An operation that needs to be synchronized with other operations that are potentially triggered by the user
- An operation that could result in losing the data entered by the user. Note that this is only relevant in non-draft scenarios
- Displaying custom messages to the user (see example)
- Changing the title of the message popup after a quick action has been performed by the system

© Copyright. All rights reserved.

```
sap.ui.controller("extsample.ext.controller.ListReportExt", {
    doSomethingREPLACE_WITH_ENTITY_SET : function(oEvent) {
        // alert('doSomerhingZCDS_UX403_OBJP_00');

        var aSelectedContexts = this.extensionAPI.getSelectedContexts();
        var oModel = this.getOwnerComponent().getModel();
        var sBindingPath = aSelectedContexts[0].getPath();

        var fnFunction = function(){
            return new Promise(function(fnResolve, fnReject){
                oModel.read(sBindingPath, {
                    success: function(oData, oResponse){
                      //do something
                        fnResolve();
                    },
                    error: function(){
                        fnReject();
                    }
                });
            })
        };

        this.extensionAPI.securedExecution(fnFunction);

    }
});
```

Figure 199: securedExecution of extensionAPI

Extending Object Page

SAP Business Application Studio provides also for the extension of the object page the guided development features. It is also possible to add a custom action, or a custom section to the object page.

Extending the object page with a custom action is comparable with extending the list report. The only difference is that the developer has to choose *Object Page* as *Page Type*.

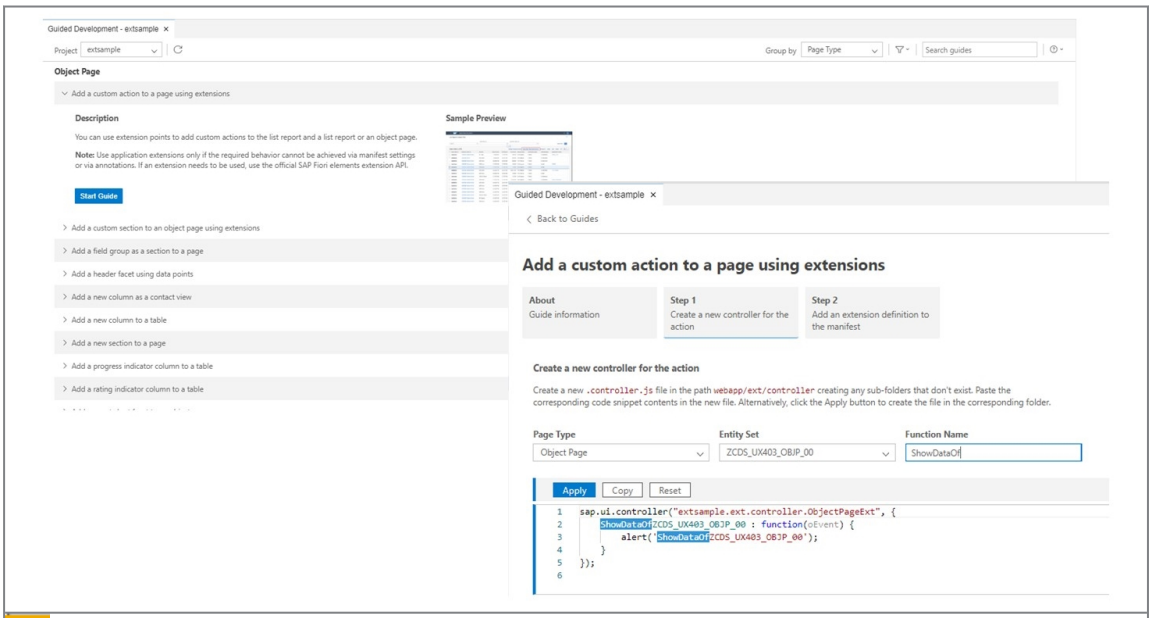


Figure 200: Extending Object Page using Guided Development

© Copyright. All rights reserved.

After the extension is generated, the extension can be implemented. SAPUI5 provides a *extensionAPI* for object page extensions.

Figure 201: Extending Object Page

As already described at the ListReport, the SAPUI5-API also provides an extension API for the object page. Details can be found at: https://ui5.sap.com/#/api/sap.suite.ui.generic.template.ObjectPage.extensionAPI

Figure 202: ObjectPage Extension API

It is also possible to add a custom section with to your object page using guided development. You can choose if you want to implement the content of the custom section using a fragment or a view. Once you have selected *apply*, the view or fragment is generated and stored in the corresponding folder.

© Copyright. All rights reserved.

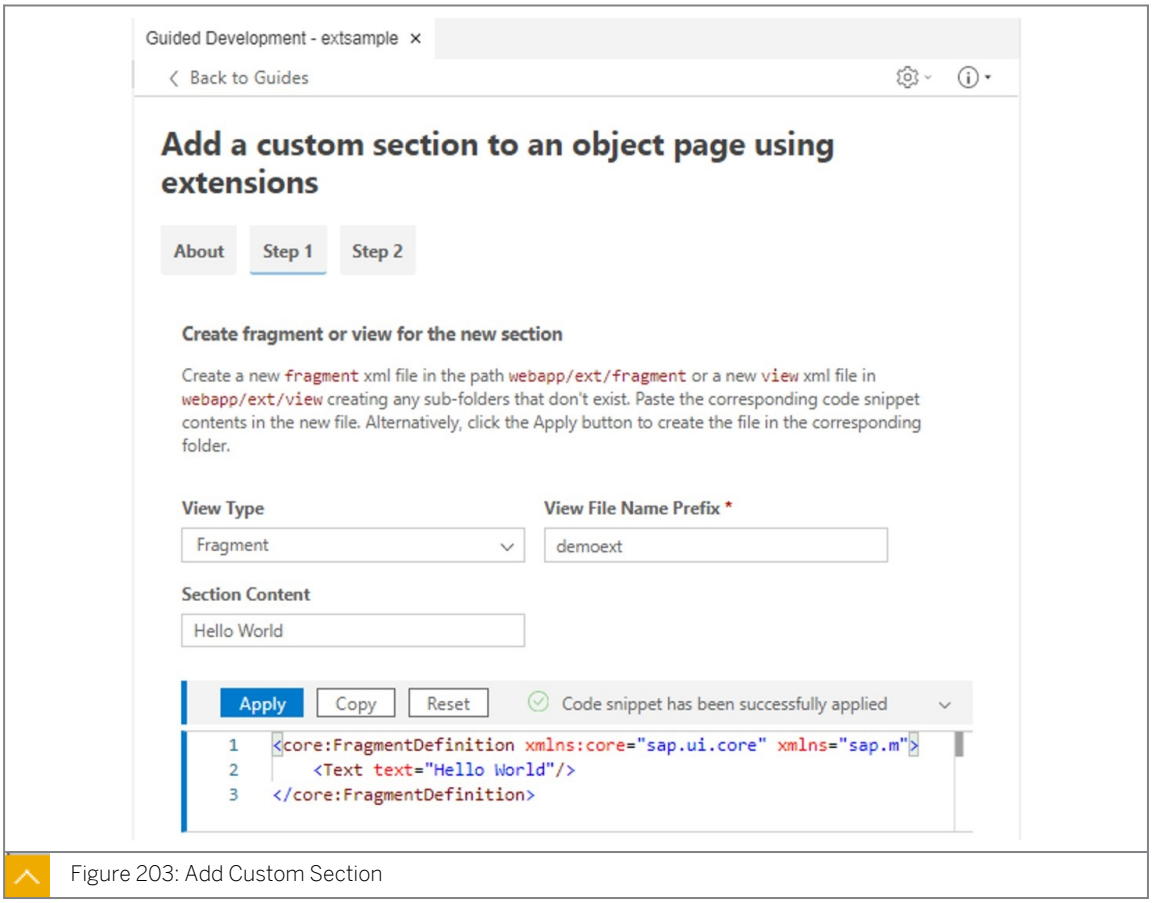


Figure 203: Add Custom Section

In the next step, you have to add the configuration to the `manifest.json` of your project. Once this is done, you can extend the generated view or fragment.

© Copyright. All rights reserved.

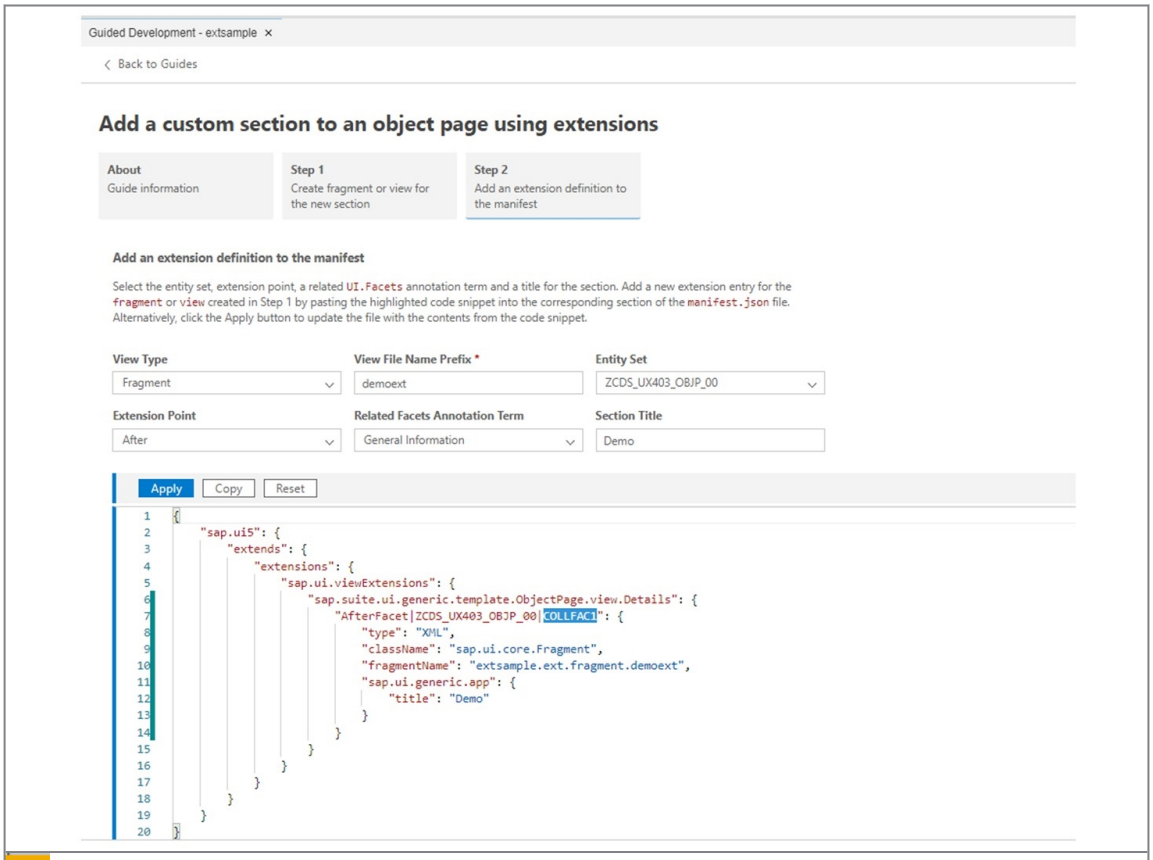


Figure 204: Add Extension Definition to Manifest

Extend Overview Page

With Guided Development, you can add a custom card and a custom filter as an extension to your overview page project. Other extensions like Custom action have to be added manually.

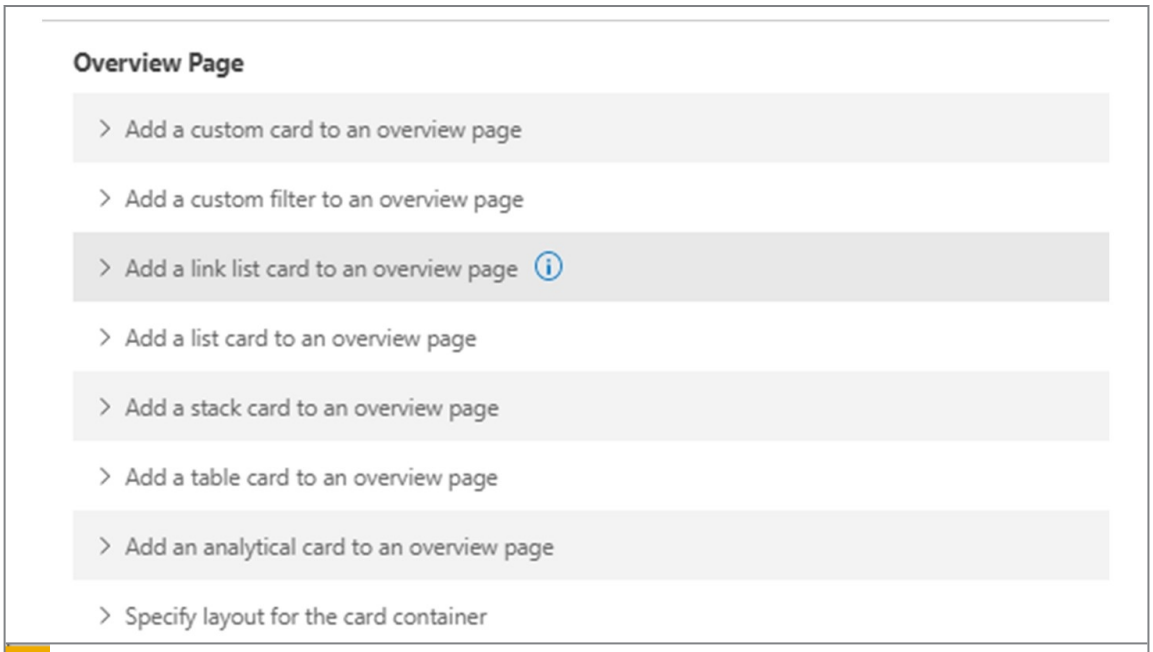


Figure 205: Extend Overview Page

© Copyright. All rights reserved.

Please refer to the documentation https://sapui5.hana.ondemand.com/1.84.1/#/topic/b240f612227547d99e7fe76dd03da375.

The following figure shows you the first step of adding a custom filter.

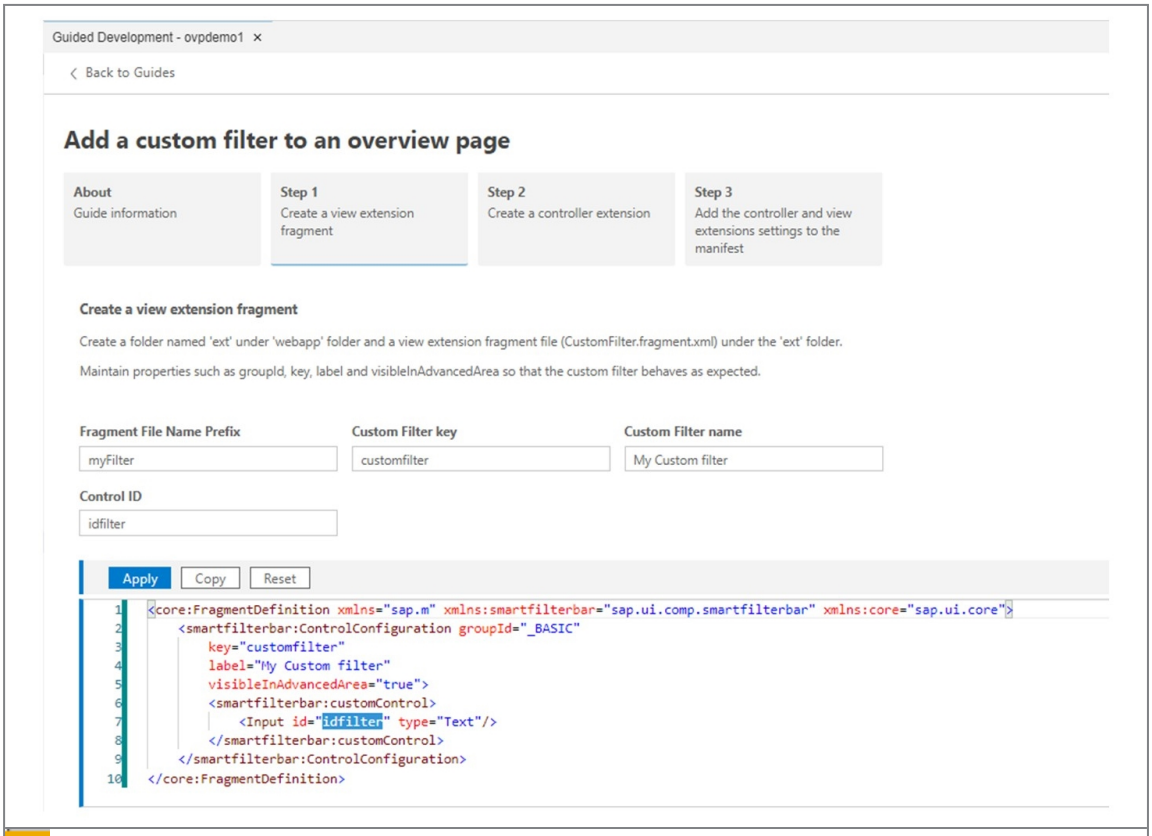


Figure 206: Adding Custom Filter - Step 1

In the second step of the guided development, the corresponding controller code is generated.

© Copyright. All rights reserved.

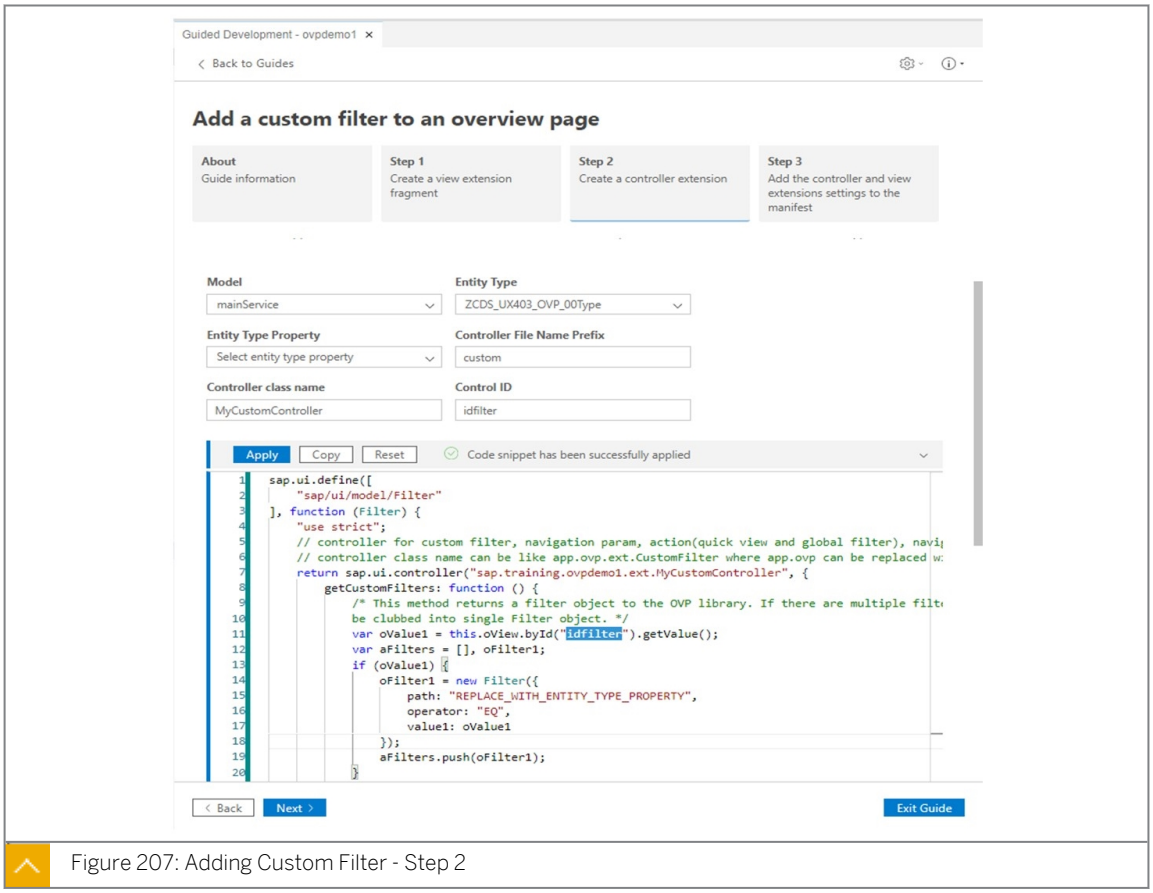


Figure 207: Adding Custom Filter - Step 2

In the third step, the configuration of the `manifest.json` is generated. The developer has to insert the name of the fragment and the name of the controller.

© Copyright. All rights reserved.

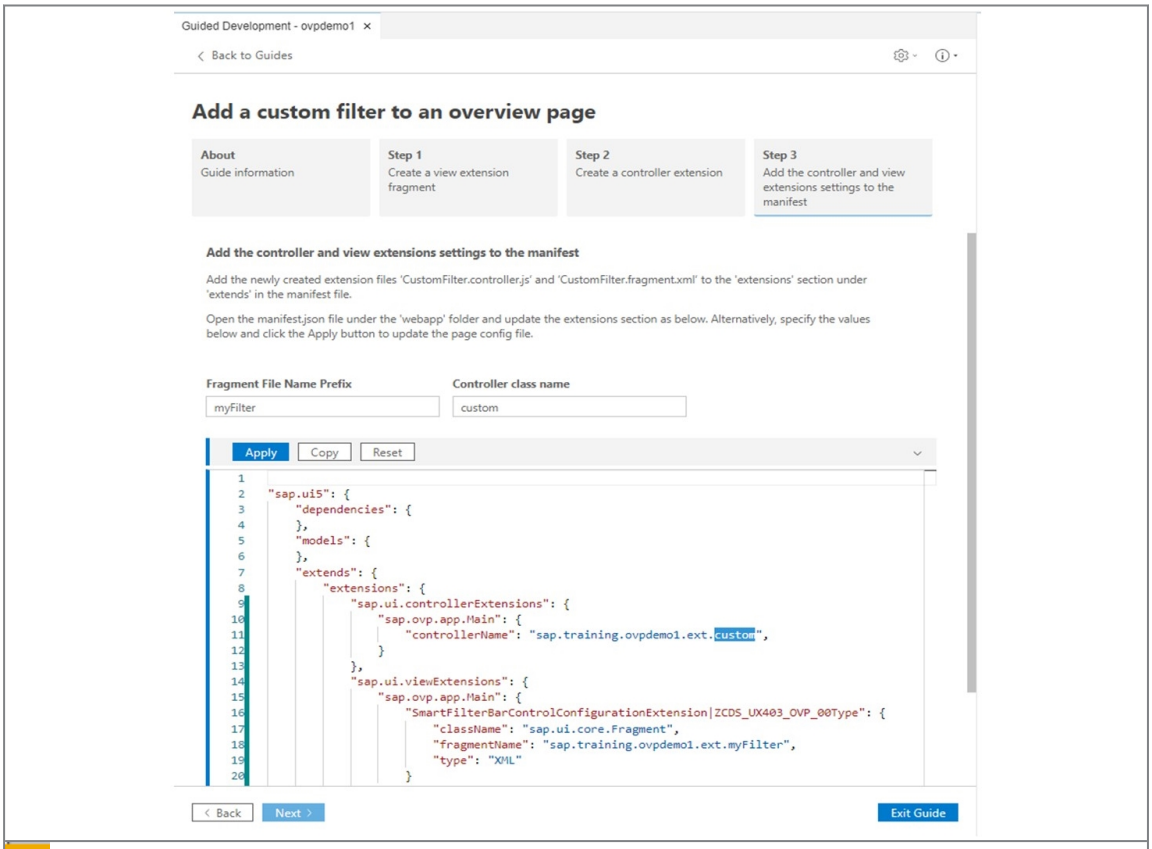


Figure 208: Adding Custom Filter - Step 3

After all necessary aspects for the custom filter are generated, the developer has to enhance the generated fragment and the generated controller to the requirements.

Figure 209: Extend the Generated Fragment to your Requirements

Extend and implement the generated functions based on your requirements:

- Use `getCustomAppStateDataExtension` to read the state of all custom fields and store that state in the object provided to enable the templates to use it for navigation.
- Use `restoreCustomAppStateDataExtension` to get the custom app state object you provided in `getCustomAppStateData` and set the corresponding values for your custom controls. For example, you call this method after returning from a navigation.

© Copyright. All rights reserved.

```
getCustomFilters: function () {
    /* This method returns a filter object to the OVP library. If there are multiple filters, they should
    be clubbed into single Filter object. */
    var sSelectedKey = this.oView.byId("CustomProductCategory").getSelectedKey();
    var aFilters = [], oFilter1, oFilter2;

    switch (sSelectedKey){
        case "All":
            break;
        case "Category1":
            oFilter1 = new Filter({
                path: "Product",
                operator: "EQ",
                value1: "HT-1000"
            });
            aFilters.push(oFilter1);

            oFilter2 = new Filter({
                path: "Product",
                operator: "EQ",
                value1: "HT-1001"
            });
            aFilters.push(oFilter2);
            break;
        case "Category2":
            oFilter1 = new Filter({
                path: "Product",
                operator: "EQ",
                value1: "HT-1002"
            });
            aFilters.push(oFilter1);

            oFilter2 = new Filter({
                path: "Product",
                operator: "EQ",
                value1: "HT-1003"
            });
            aFilters.push(oFilter2);
            break;
        default:
            break;
    }

    if (aFilters && aFilters.length > 0) {
        return (new Filter(aFilters, false));
    }
},

getCustomAppStateDataExtension: function (oCustomData) {
    //the content of the custom field will be stored in the app state, so that it can
    //be restored later, for example after a back navigation.
    //The developer has to ensure that the content of the field is stored
    //in the object that is returned by this method.
      if (oCustomData) {
        var oCustomField1 = this.oView.byId("Product");
        if (oCustomField1) {
            oCustomData.Product = oCustomField1.getValue();
        }
    }
},

restoreCustomAppStateDataExtension: function (oCustomData) {
    //in order to restore the content of the custom field in the
    //filter bar, for example after a back navigation,
    //an object with the content is handed over to this method and
    //the developer has to ensure that the content of the custom field is set accordingly
    //also, empty properties have to be set
    if (oCustomData) {
        if (oCustomData.Product) {
            var oCustomField1 = this.oView.byId("Product");
            oCustomField1.setValue(oCustomData.Product);
        }
    }
}
```

Figure 210: Extend the Generated Controller to your Requirements

Create a Custom Card

As mentioned in the OVP- unit customers can implement their own cards. Therefore, the SAP Business Application studio provides a guided development that guides developers to the necessary steps.

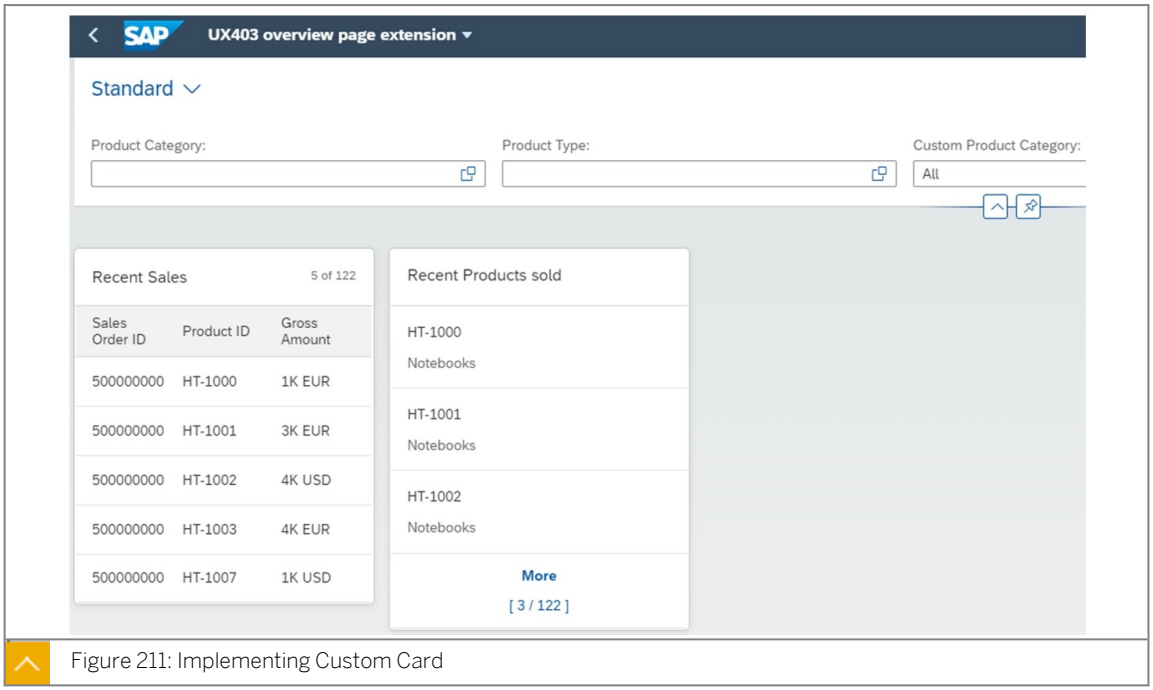


Figure 211: Implementing Custom Card

© Copyright. All rights reserved.

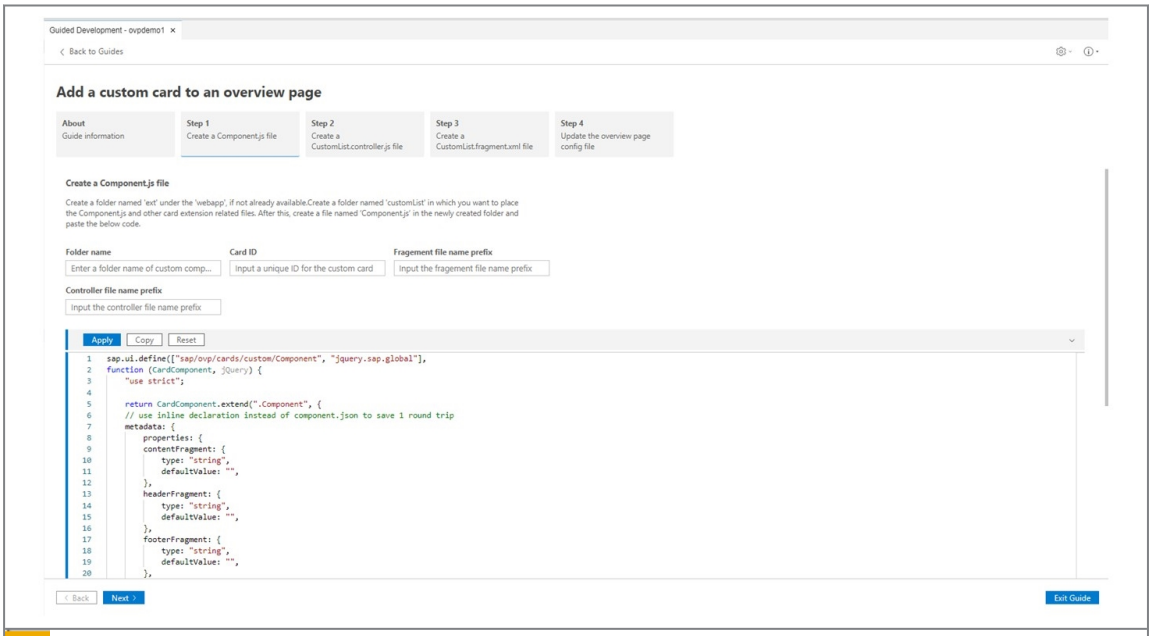


Figure 212: Create a Custom Card with Guided Development

A custom card is implemented as a self-containing component.

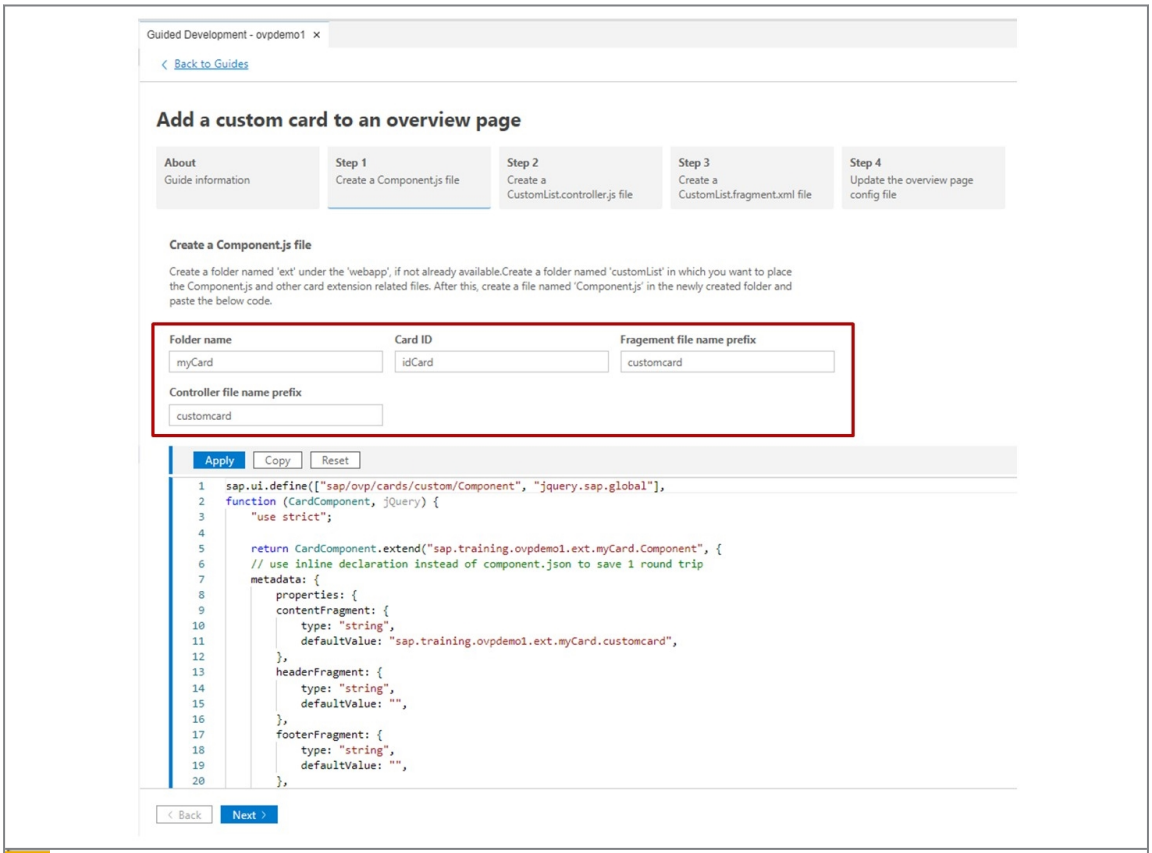


Figure 213: Custom Card – Step 1

© Copyright. All rights reserved.

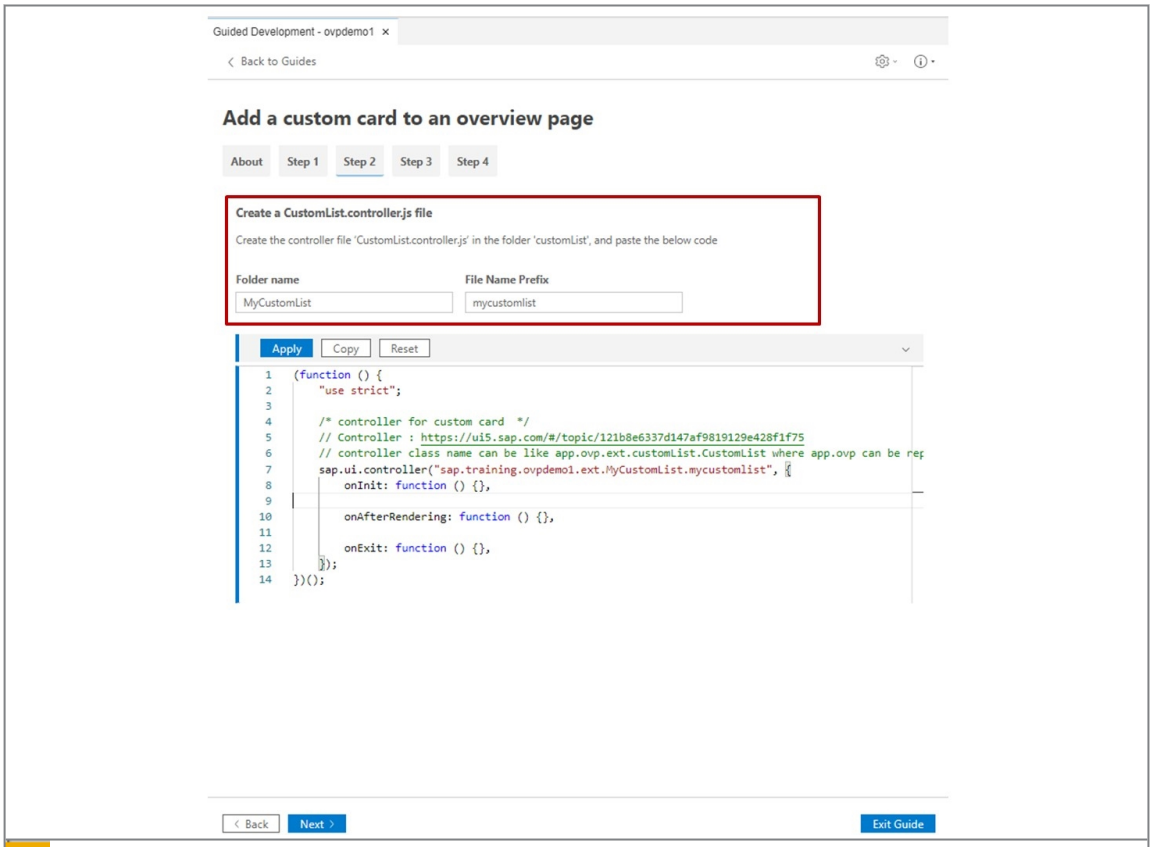


Figure 214: Custom Card – Step 2

In the third step, a fragment is configured that contains the card implementation.

© Copyright. All rights reserved.

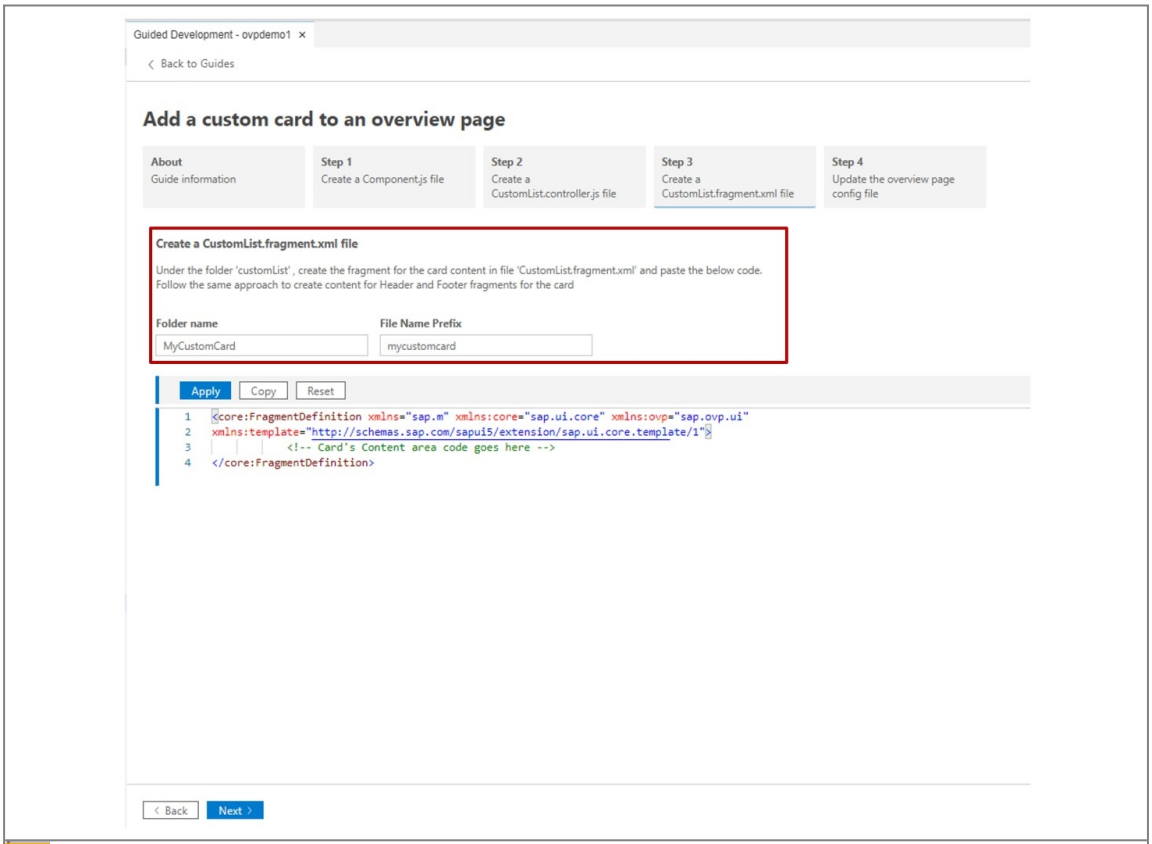


Figure 215: Custom Card – Step 3

In the final step, the configuration is added to the `manifest.json`. You have to insert the name of the component and the id of the card that was inserted in Step 1 of the guided development.

© Copyright. All rights reserved.

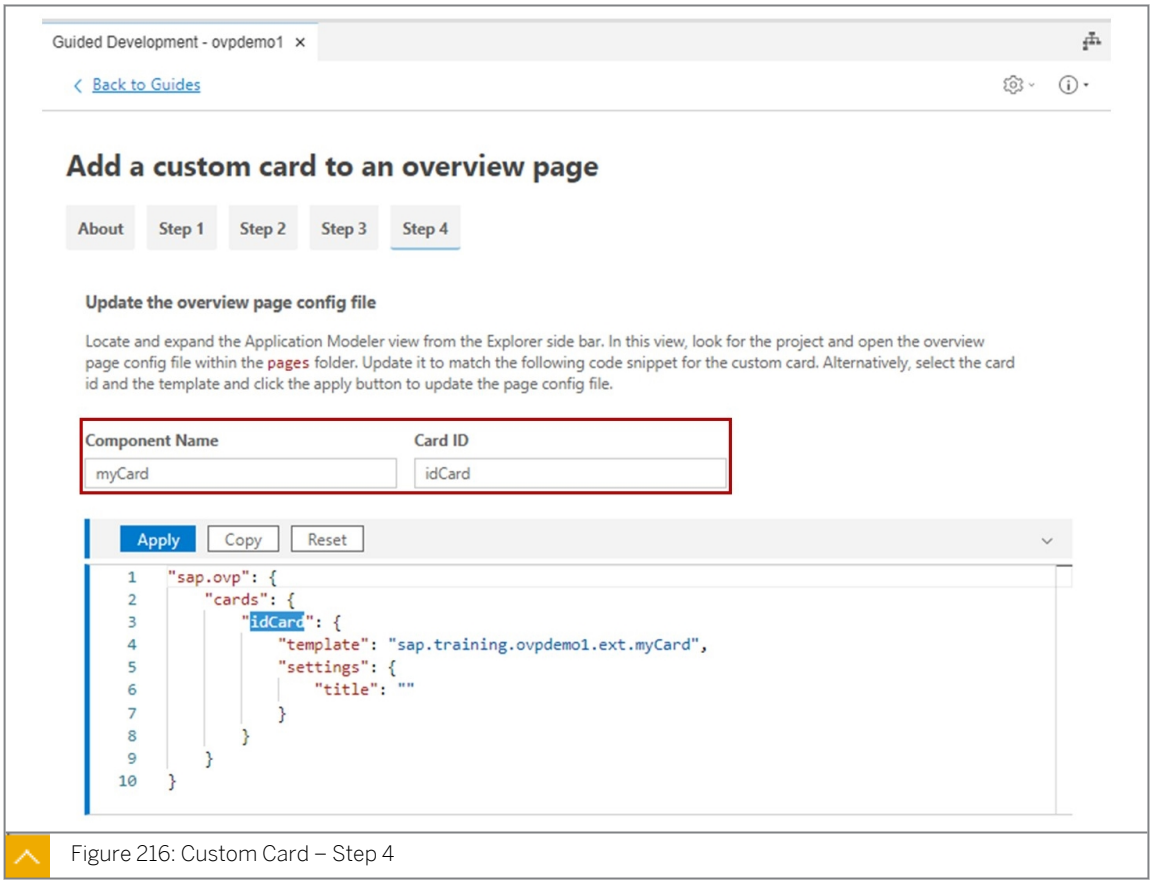


Figure 216: Custom Card – Step 4

After the code artefacts are generated, you can implement the card. You have to implement the fragment for the card header and the fragment for the card content.

Figure 217: Extend the Generated Artefacts

SAPUI5 Flexibility

Introduction

- SAPUI5 flexibility enables functions for different user groups to adapt SAPUI5 applications in a simple and modification-free way.

© Copyright. All rights reserved.

- It is available on ABAP platform, SAP Cloud Platform services in the Cloud Foundry environment.
- It replaces the extensibility concept by broadening the adaptability of SAPUI5 application and simultaneous increase of maintainability and simplicity.

Figure 218: SAPUI5 Flexibility

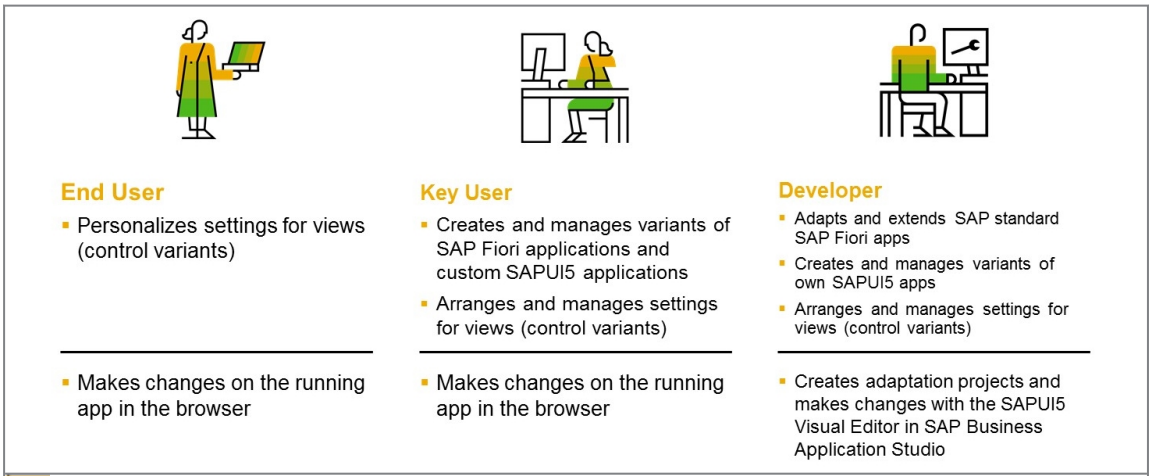


Figure 219: Personas, Use Cases, and Tools

UI adaptation is a feature of SAPUI5 flexibility that allows key users without technical knowledge and developers to easily make UI changes in a what you see is what you get (WYSIWYG) manner.

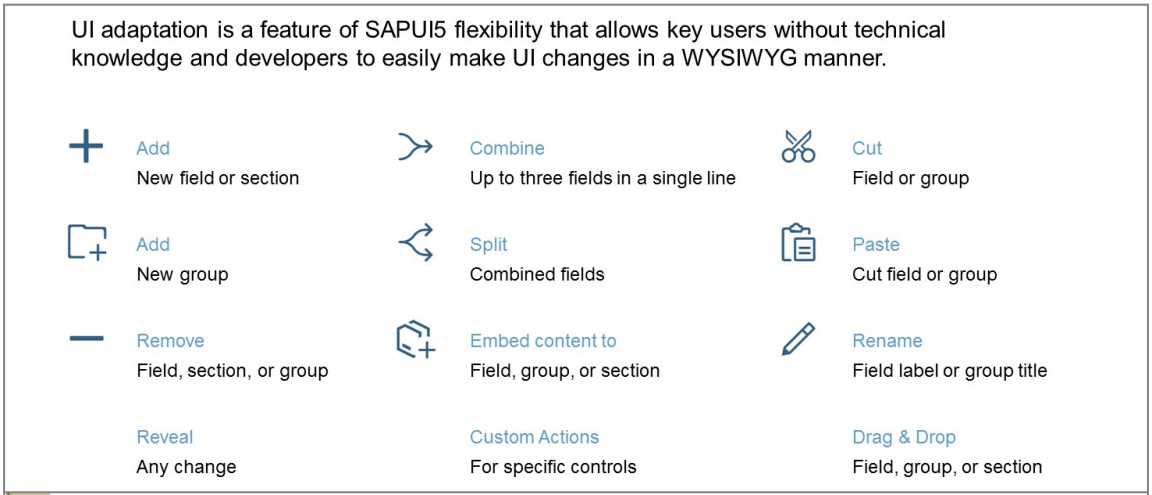


Figure 220: UI Adaption for Developers and Key Users

© Copyright. All rights reserved.

SAPUI5 flexibility allows UI adjustments by creating app variants from existing applications. The UI of the applications can therefore be adapted separately without touching the original app.

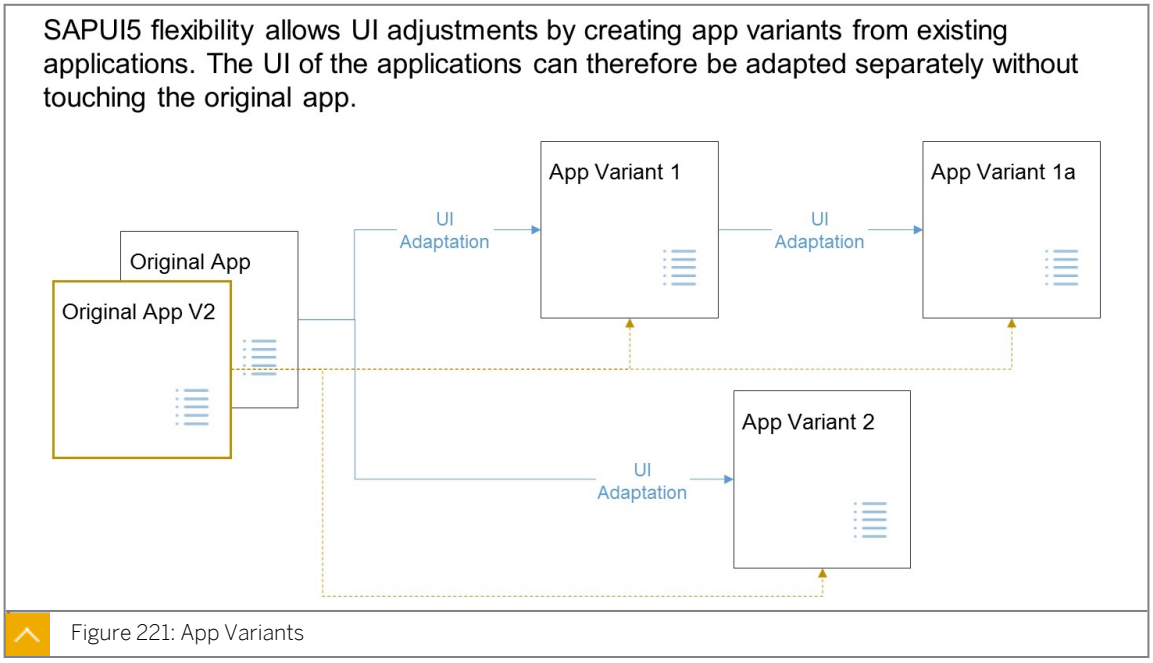


Figure 221: App Variants

Further details on the App Variant concept can be found at https://help.sap.com/viewer/a7b390faab1140c087b8926571e942b7/201809.002/en-US/af47058ad66144579db6a990f3b7b919.html

Adaption Project

SAPUI5 Adaptation Project allows developers to extend SAP Fiori application in SAP Business Application Studio.

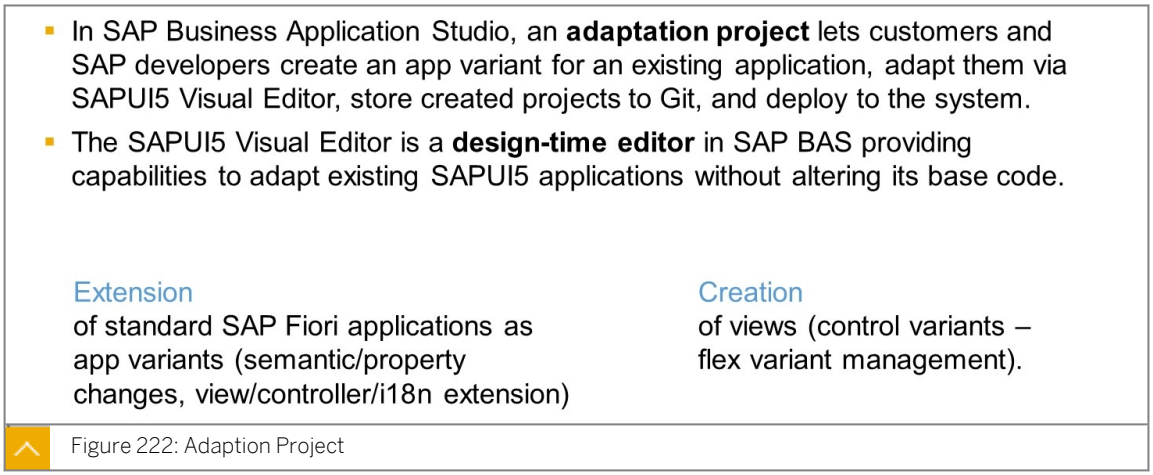


Figure 222: Adaption Project

SAPUI5 Adaptation Project lets you create an app variant for an existing SAP Fiori elements-based application on SAP S/4HANA on-premise ABAP system and provides extension capabilities for UI5 controls. Create an app variant that includes the changes that you make to the source application and to the variant itself.

Extensions are made via SAPUI5 flexibility possibilities and do not modify the source. This modification-free approach allows you to reference the source application and its artifacts instead of modifying the source artifacts itself. An adaptation project behaves as follow –

© Copyright. All rights reserved.

start as a preview, adapt using SAPUI5 Adaptation Project, and deploy from SAP Business Application Studio. The artifacts on an adaptation project represent only the changes that you make and not the entire application that you reuse.

The app variant refers to the original application, but contains a separate set of changes created in the adaptation project. Also, an application ID is defined for the variant and needs separate registration in SAP Fiori Launchpad. The option to create app variants based on existing apps allows you to keep both instances running and keeps the original application untouched. You can configure both application instances as different tiles on FLP. You can work with both applications, or assign the app variant to a different set of users.

Currently, adaptation projects are supported only on on-premise ABAP systems with the minimum required version of software component being SAP_UI 7.53 SP00. Keep in mind that, if you have SAP_UI 7.53 SP00 installed, you need to apply the SAP note 2615176.

The minimum SAPUI5 version must be 1.71.

The SAP Fiori App Reference Library gives information what extensibility features are provided and what technical application name stands behind the application name. The SAP Guided Answers provide detailed information on how to extend. Further information on extensibility can be found at https://wiki.scn.sap.com/wiki/display/Fiori/Extensibility.

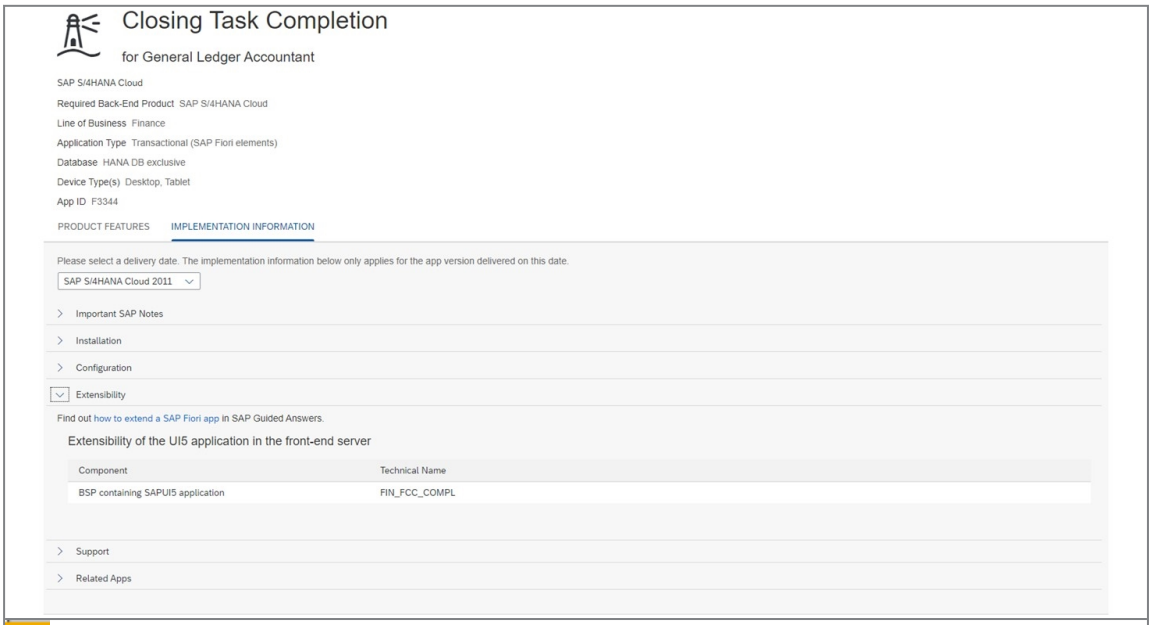


Figure 223: SAP Fiori App Reference Library

Adaptation Project allows developers to extend SAP Fiori application in SAP Business Application Studio. For using all features, it is important that the base application provides stable Ids on the level of controls. SAP Fiori elements makes sure that each control has a stable Id. After the adaption project is generated, the new project has to be opened in a new workspace. If not the Visual Editor is not working correctly (this is the current state and might change in the future).

© Copyright. All rights reserved.

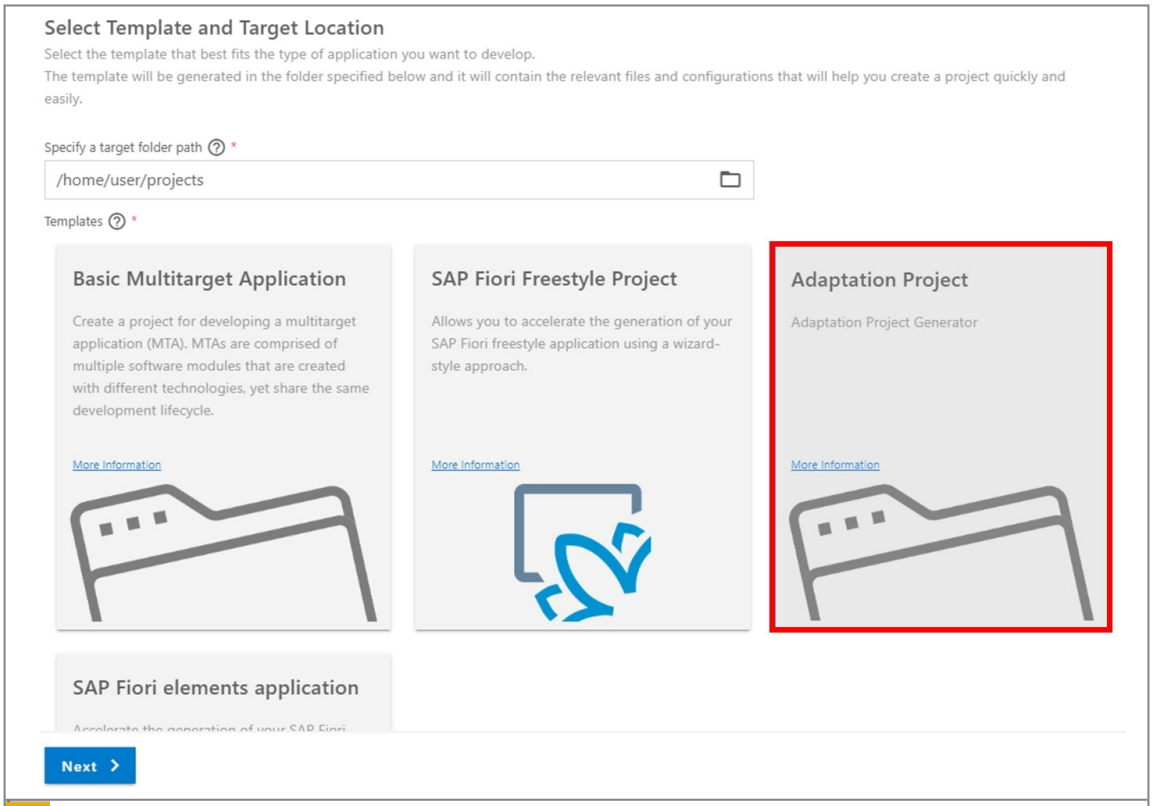


Figure 224: Adaptation Project

The SAPUI5 Visual Editor is used to extent the base application by creating an application variant.

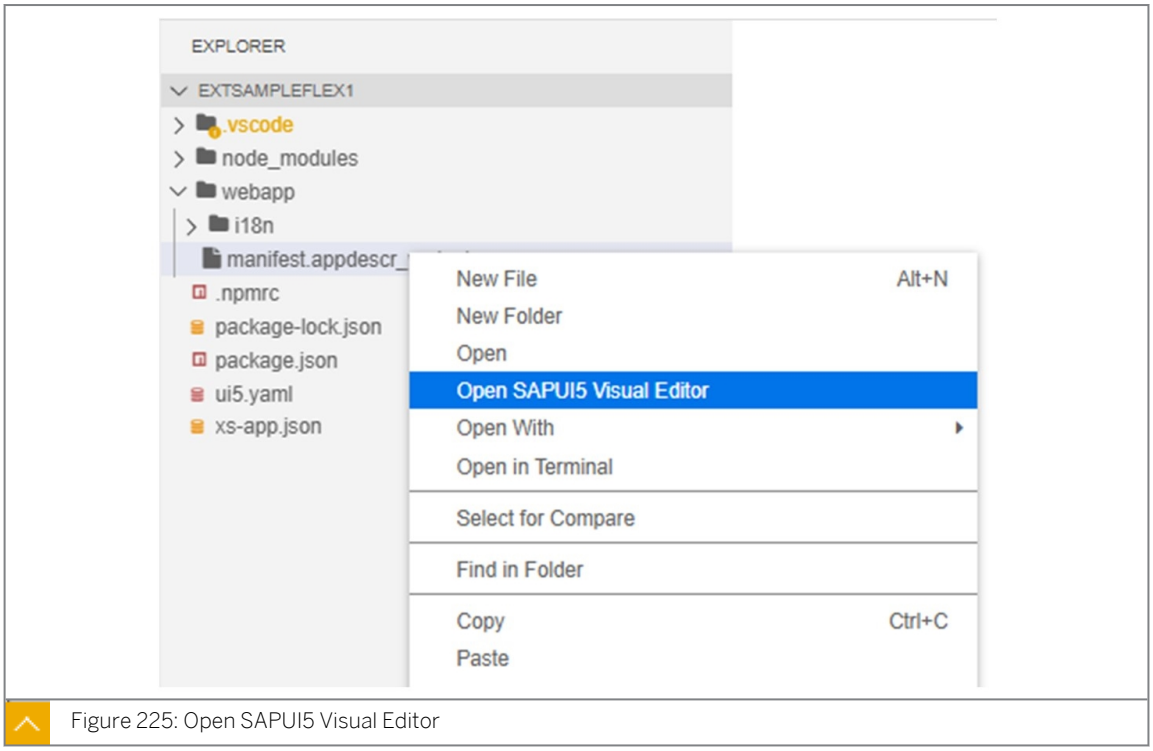


Figure 225: Open SAPUI5 Visual Editor

In the *Edit* mode the developer can adapt the current implementation. It is important that the base application uses stable Ids. Without stable IDs. For example, the properties of existing

© Copyright. All rights reserved.

controls are not editable. Using the funnel symbol, the developer can enhance/reduce the control tree in the outline view and display all controls available in the base application.

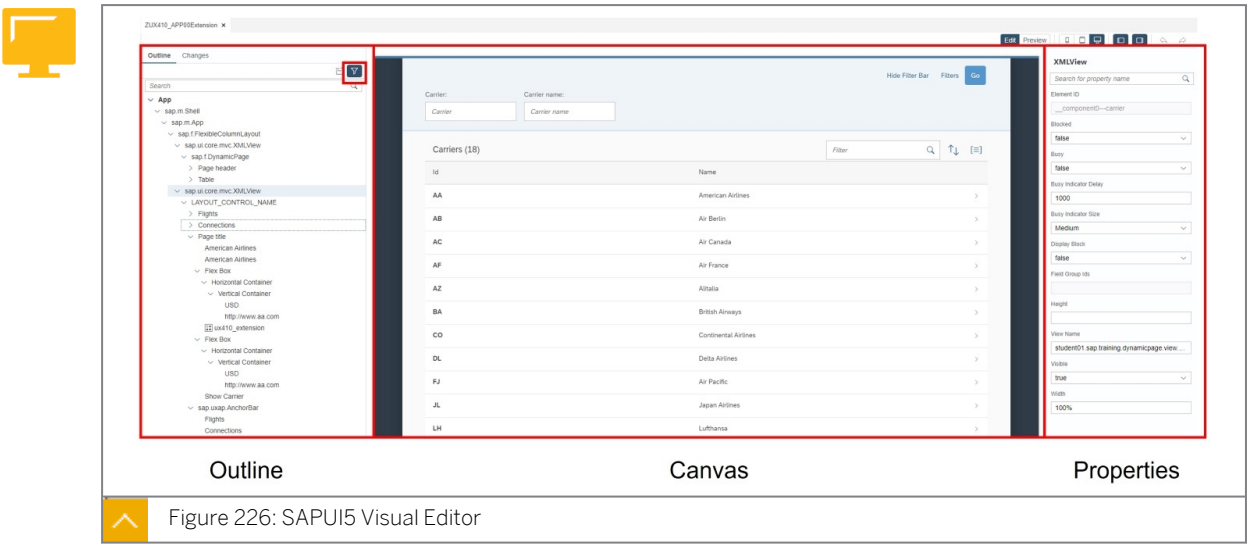


Figure 226: SAPUI5 Visual Editor

You can start the visual editor, which will enable you to both preview and edit the application. The possibilities are the same for both SAP Fiori elements-based and Freestyle applications and you can extend them in the same manner – you can adapt UIs, for example add/move/remove fields, rearrange sections or even embed iframes, change control properties, extend i18n texts, add custom content via XML Fragments, and so on. You can also extend controllers by using the right-click option *Extend with controller*.

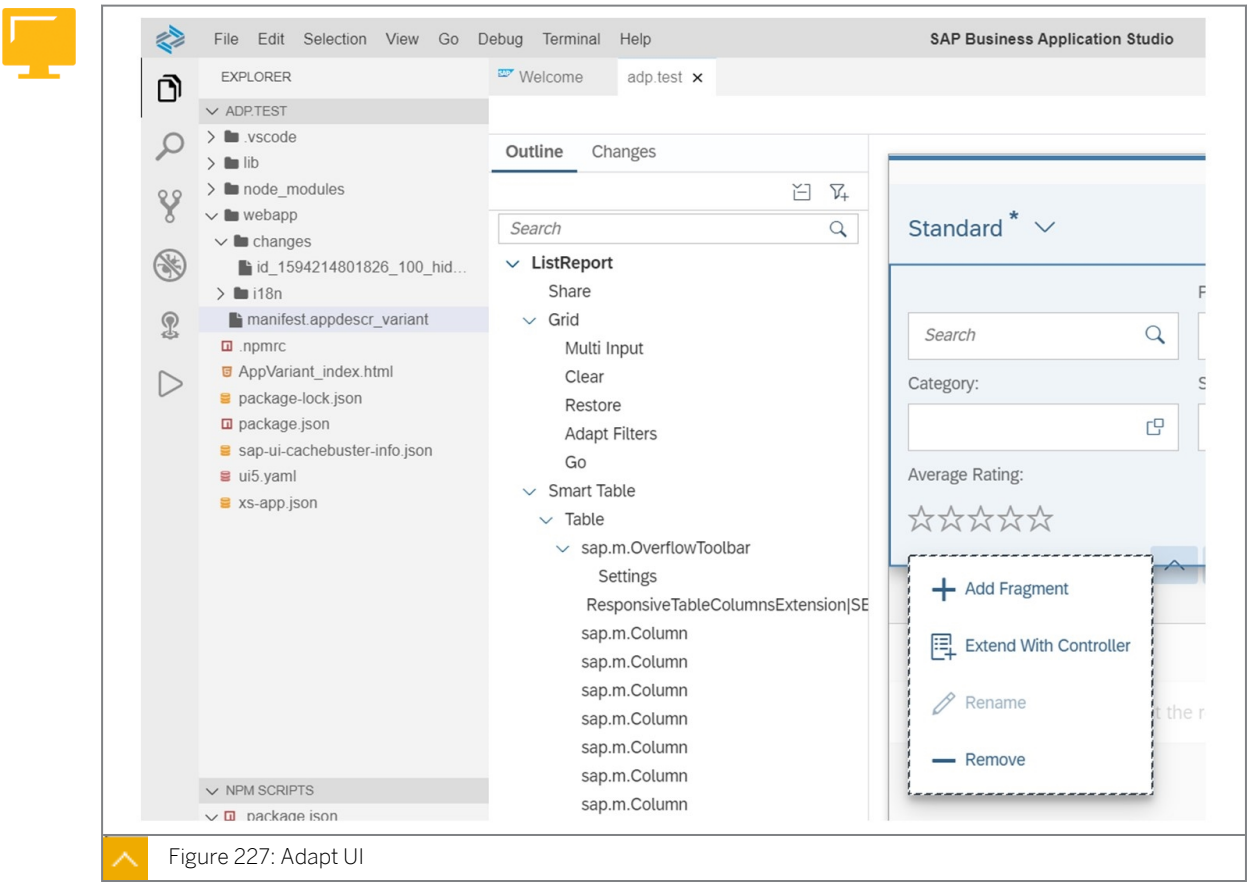


Figure 227: Adapt UI

© Copyright. All rights reserved.

Adaptions added or configured in the visual editor are stored *.change files*. The development artefacts are stored in the changes folder located in the project. Controller-extensions are stored in the coding subfolder, view enhancements are stored in the fragments subfolder. For view modifications, no SAPUI5-code is generated; only a *.change* file is created.

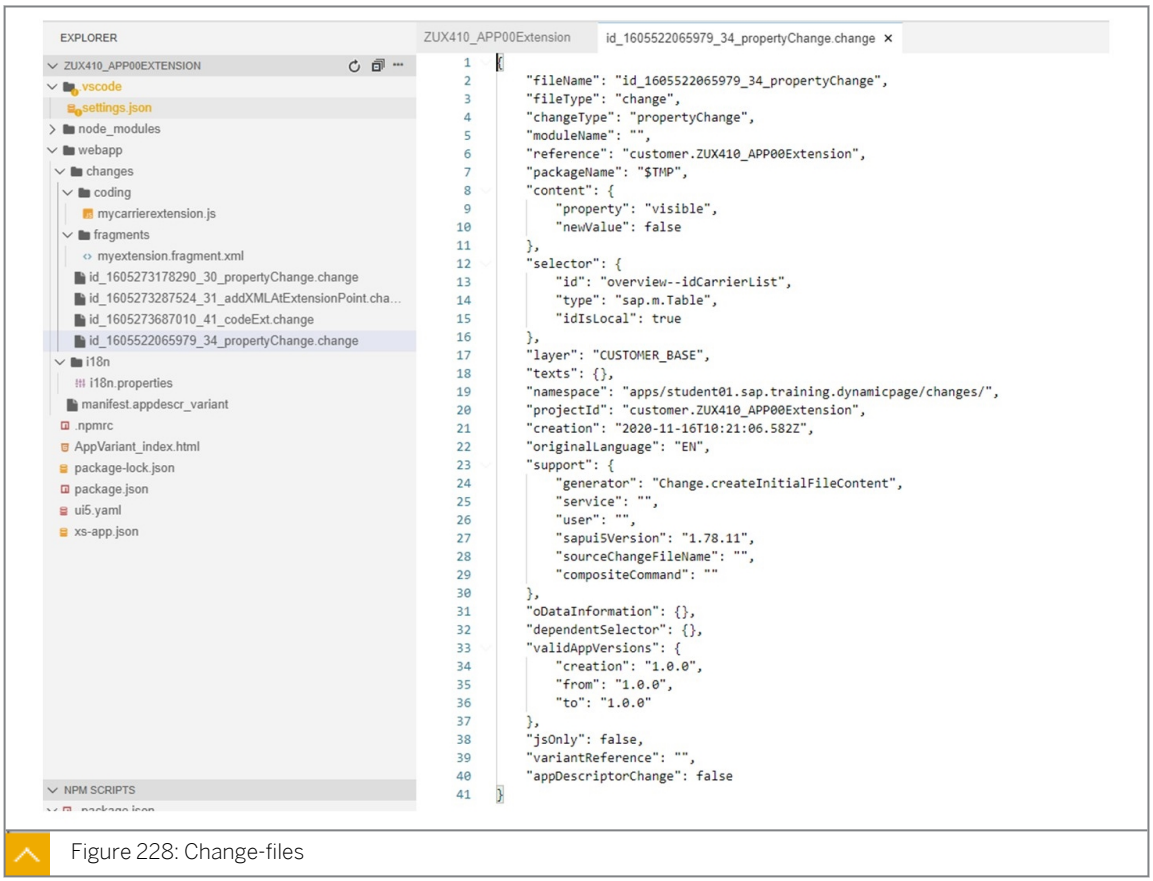


Figure 228: Change-files

Use the *yo* tool to deploy a new application variant use as `namespace` parameter the namespace specified in the file *manifest.appdescr_variant* of your adaption project.

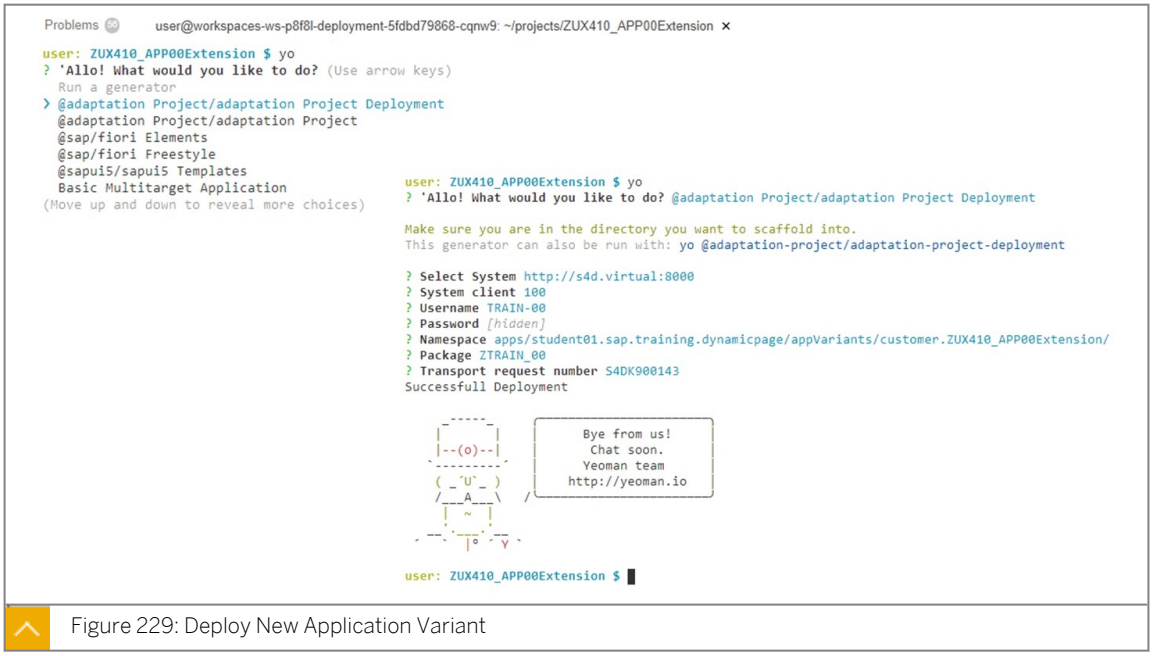


Figure 229: Deploy New Application Variant

© Copyright. All rights reserved.

Access the Layered Repository using transaction `SUI_SUPPORT`.

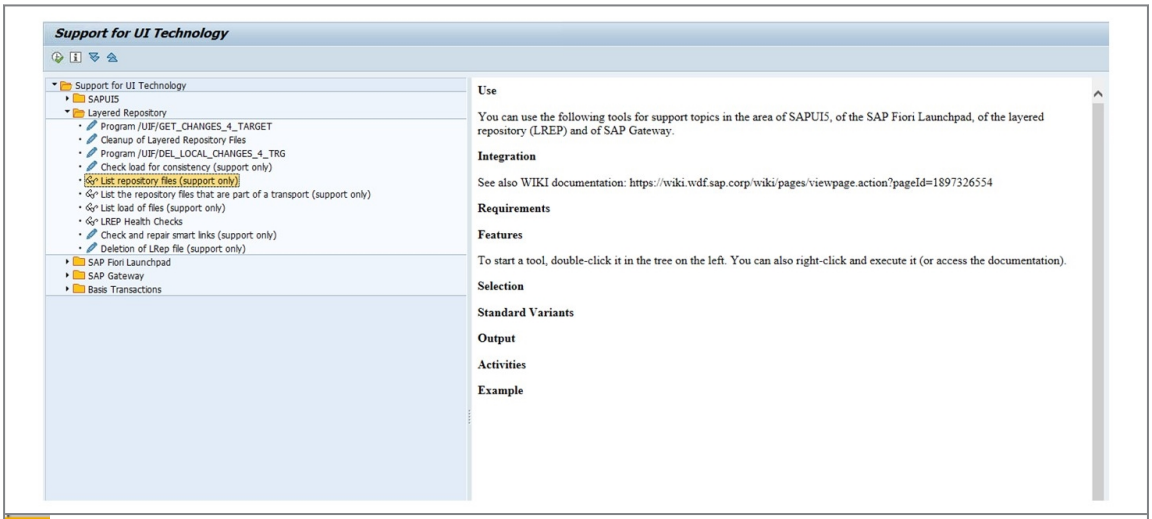


Figure 230: Transaction SUI_SUPPORT

Having this detail, the administrator can now create a new target mapping to the application variant. If you want to hide the base application in general, add the URL parameter *sap-appvar-id* with the namespace of your application variant to the target mapping of the base application.

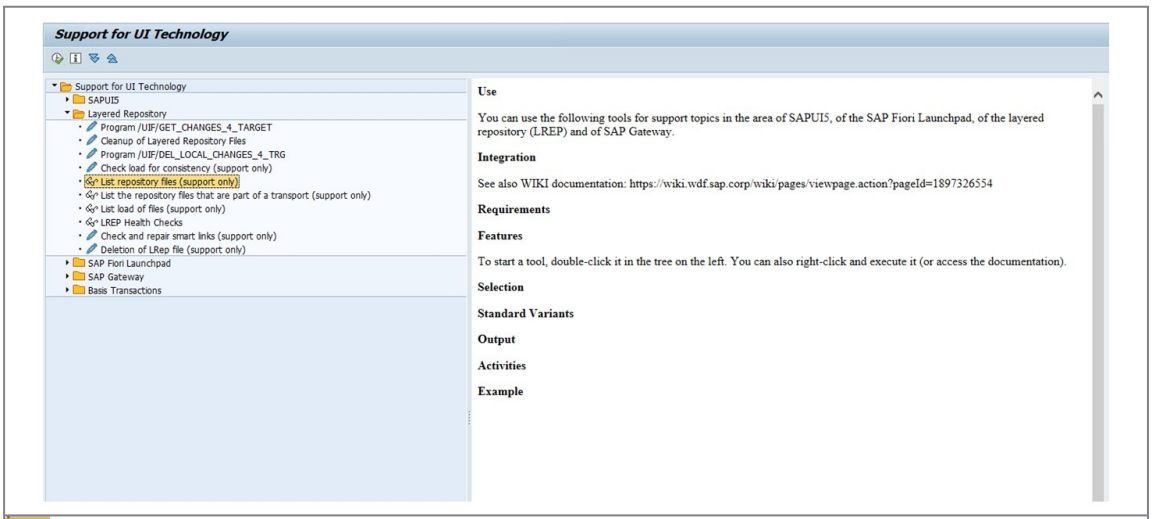


Figure 231: Details for the Deployed Application Variant

LESSON SUMMARY

You should now be able to:

- Extending and Adapting SAP Fiori Elements Applications

© Copyright. All rights reserved.