

HANDBOOK ABAP Development Tools (ADT) in Eclipse

As of March 2023

Table of Contents

Table	e of C	Conten	ıts	2
Table	e of F	igures	\$	6
Inde	x of T	ables		12
Intro	ducti	on		13
	Why	a DSA	G guide on ABAP Development Tools (ADT) in Eclipse?	13
	Motiv	ation f	or and topicality of the guide	13
1	Intro	ductio	on Eclipse and ADT	15
	1.1	Eclips	e Why?	15
	1.2	A brie	f look at the history of its creation	15
	1.3	The co	oncept of extensions	15
	1.4	Differe	ance between perspective and view	16
	1.5	The po	ower of keyboard shortcuts	16
	1.6	Which	Eclipse feature is available in which version?	16
2	Moti	vation	for ADT	18
	2.1	You w	ant to apply and implement modern ABAP	18
	2.2	You w	ant to use one development environment for everything	18
		2.2.1	Technological aspect	18
		2.2.2	Development on multiple system lines	19
	2.3	You w	ant to break free from the technical limitations of the SAP GUI	19
		2.3.1	Six sessions in one system are simply not enough	19
		2.3.2	You don't want to lose code due to network interruptions	20
	2.4	You lil	ke to improve your ABAP code through refactoring	20
	2.5	You lil	ke to adapt the look and feel of your development environment to	
		your w	vishes	21
	2.6	Would	I you like even more tool-based support?	21
	2.7	You a	Iready know Eclipse from other programming languages? Great!	22
	2.8	You a	re interested in new things	22
	2.9	The be	enefits for the organization	23
	2.10	You a	re missing the form-based editor	23
	2.11	Why A	BAP Development Tools	24
3	Worl	king w	ith ADT	26
	3.1	Introdu	uction: Basics of working with ADT	26
		3.1.1	Getting Started with ABAP Development Tools	26
		3.1.2	The shift from form-based to text-oriented code creation	27

		3.1.3	Context of the exercise	. 27
		3.1.4	Connecting the Development System – New Project	. 28
		3.1.5	Creating a class in text mode	. 30
		3.1.6	Definition of a method in the class	. 33
		3.1.7	Automatic completion and formatting of the code	. 35
		3.1.8	Implementation of the method using Quick Fix	. 36
		3.1.9	Renaming of Parameters – Refactoring	. 39
	3.2	Featu	res of ADT	. 43
		3.2.1	Overarching features	. 43
		3.2.2	Search and Navigation	. 51
		3.2.3	ABAP Editor	. 55
		3.2.4	Other object types	. 61
		3.2.5	ABAP Views	. 62
		3.2.6	Refactoring Code with ADT	. 88
		3.2.7	Versioning and Comparison	. 90
		3.2.8	Documentation with ABAP Doc	. 92
		3.2.9	Executing Source Code	. 94
		3.2.10	Data Preview	. 97
		3.2.11	Core Data Services	100
4	Τιοι	ublesh	ooting Tools in Eclipse	107
	4.1	Troub	leshooting with the ABAP Development Tools	107
	4.2	The D	Debugger in ABAP Development Tools	108
		4.2.1	Breakpoints and Soft-Breakpoints	108
		4.2.2	Debugging Perspective	108
		4.2.3	Special Behavior in the Debugger	109
		4.2.4	Additional Information	109
	4.3	Check	kpoint IDs and dynamic logpoints	110
	4.4	Perfo	rmance Analysis	113
		4.4.1	Note on HANA Studio and SAP HANA Tools	123
	4.5	Feed	Reader	123
_	4.6	Docur	mentation Links	126
5	Inst		n, Distribution und Update Strategies	127
	5.1		Installation Guide of SAP	127 107
		5.1.1	Installation Guide of SAF	121

	5.2	rations	27	
		5.2.1	Java Development Kit and Java Runtime Environment	27
		5.2.2	Backend12	28
		5.2.3	SAP GUI Installation12	29
		5.2.4	Visual Studio Redistributable12	29
	5.3	Techr	nical Structure of an Eclipse Installation12	29
	5.4	Plug-i	ns 13	30
		5.4.1	Eclipse Marketplace1	30
		5.4.2	Update Site 13	32
	5.5	Install	ation and Distribution Strategies1	34
		5.5.1	Overview and Comparison13	34
		5.5.2	Manual Installation 13	36
		5.5.3	Pre-configured Initial Installation13	36
		5.5.4	Eclipse Installer1	37
	5.6	Error	Situations1	54
		5.6.1	Errors during update or upgrade 15	54
		5.6.2	Single-sign-on Libraries	55
		5.6.3	"No repository found containing"1	55
		5.6.4	PKIX - Certificate Error	55
		5.6.5	macOS aarch64 support & SAP GUI for Java	56
		5.6.6	Offline-Installation – Download ADT-Dependencies	56
6	Best	Pract	ices Eclipse Configuration1	57
	6.1	Settin	gs in Eclipse	57
		6.1.1	Globale Settings	57
		6.1.2	Project-specific Settings10	63
	6.2	Views	and Perspectives	65
		6.2.1	Views	65
		6.2.2	Perspectives	68
	6.3	Recor	nmended additional views1	70
	6.4	Sugge	estions for distribution1	71
7	Plug	-ins		72
	7.1	Introd	uction1	72
	7.2	Usefu	I Open-Source Plug-ins1	72
		7.2.1	Open Editors	72

	7.2.2	AnyEdit Tools	173			
	7.2.3	PDE Tools	175			
7.3	7.3 Useful Open-Source ADT Plug-ins					
	7.3.1	ABAP Favorites	176			
	7.3.2	ABAP Continuous Integration	178			
	7.3.3	ABAP ADT Extensions	179			
	7.3.4	ADT Classic Outline	182			
	7.3.5	ABAP Quick Fix	183			
	7.3.6	ABAPQuickFixS4Conversion	184			
	7.3.7	ABAP Tags	186			
	7.3.8	ABAP Search and Analysis Tools	188			
	7.3.9	ABAP Code Search	189			
	7.3.10	DabapGit Eclipse Plug-in	191			
7.4	Devel	lop own ADT Plug-ins	191			
	7.4.1	Prerequisites	191			
	7.4.2	Setting up the development environment	192			
	7.4.3	Key Concepts/Artifacts	194			
	7.4.4	Creation of a Plug-in Project	194			
	7.4.5	Creation of a Feature Project	198			
	7.4.6	Creation of an Update Site	200			
	7.4.7	Extension of the ADT Backends with ABAP Code	203			
	7.4.8	Java Code Snippets for recurring tasks in ADT	205			
	7.4.9	Project Set-up with Maven	209			
Authors			210			
Imprint.			214			



Table of Figures

Figure 1 Creating an ABAP Project in Eclipse	28
Figure 2 The Project Explorer	29
Figure 3 Adding Packages to Favorites	30
Figure 4 Creating a New ABAP Class in Project Explorer	31
Figure 5 Properties Dialog: Creating an ABAP Class	31
Figure 6 Transport Request Dialog	32
Figure 7 Display of the new class in ADT	33
Figure 8 Editing the Class	34
Figure 9 Example Code Completion for the Import Parameter	35
Figure 10 Using the Quick Fix for Method Implementation	37
Figure 11 Component Selection Using Code Completion	38
Figure 12 Renaming Method Parameters	40
Figure 13 Query Dialog for saving the Code	40
Figure 14 Entering a new Parameter Name	41
Figure 15 Transport Selection and Options	41
Figure 16 Renaming Preview	42
Figure 17 Query of the Workspace Directory	43
Figure 18 Switching the Workspace	45
Figure 19 The Workspace Dialog	46
Figure 20 Adding Packages to Favorites	47
Figure 21 Project Explorer Detail Screen with Button Bar	48
Figure 22 Working Sets Settings	49
Figure 23 Creation and Processing of the Working Sets	49
Figure 24 Assignment of Project to Working Set	50
Figure 25 Project Explorer Display Setting	50
Figure 26 Illustration of Projects in Working Sets	51
Figure 27 Object Search Dialog	52
Figure 28 Results of the Search	53
Figure 29 Display of additional Search Options	53
Figure 30 Result with Object and Type Filter	53
Figure 31 Navigations Icons	54

Figure 32	Actions for the Project Explorer	55
Figure 33	ABAP Editor - Main Window	56
Figure 34	Element Info for a Method	56
Figure 35	Element Info for a Data Element	57
Figure 36	Display of the Element Info after Selecting the Object	58
Figure 37	Formatting Context Menu	59
Figure 38	Settings for the ABAP Formatter	60
Figure 39	Display of Refactoring Options	61
Figure 40	Comparison of Function Groups in the Project Explorer of ADT and in the SE80	е 61
Figure 41	Display of Properties in the Outlines	62
Figure 42	Displaying the Messages in the Problems View	63
Figure 43	Display of the Options of the View	63
Figure 44	Configuration of the Displayed Points	64
Figure 45	Properties View	65
Figure 46	History of Transports	65
Figure 47	Transport View	66
Figure 48	Template View Browser	67
Figure 49	Selection of Templates in Content Assist	68
Figure 50	Result of the Proof of Use/Where-used List	69
Figure 51	Filter for Where-Used-Search	70
Figure 52	Menu for Creating the Bookmark	71
Figure 53	Entering a Name (Bookmark)	71
Figure 54	Display of a Bookmark in the Source Code	72
Figure 55	Bookmarks View	72
Figure 56	Sharing the Source Code as a Link (Context Menu)	73
Figure 57	Link Sharing Dialog	74
Figure 58	Opening the ABAP Type Hierarchy	75
Figure 59	Display of the Type Hierarchy in the View	75
Figure 60	Transport Organizer View	76
Figure 61	Representation of a Run-Time Error	76
Figure 62	Example of a System Message	77
Figure 63	Executing the ABAP Unit Test via the Context Menu	77

Figure 64	Display of the Results of the ABAP Unit Test	78
Figure 65	Restart the Execution	78
Figure 66	Dialog for Setting the Execution of ABAP Unit Tests	79
Figure 67	Carrying out the Coverage Measurement	79
Figure 68	Color Highlighting of Source Code after Unit Test	80
Figure 69	Display of the Results of the ATC Run	81
Figure 70	Requesting Exemptions via the ATC View	81
Figure 71	Dialog for Classifying the Exception	82
Figure 72	Calling the ABAP Language Help from the Context Menu	83
Figure 73	ABAP Language Help View	84
Figure 74	Button Bar of the View	84
Figure 75	Navigation to Help Content	85
Figure 76	Overview of Available Help and Documentation	86
Figure 77	Search in Help	87
Figure 78	Further Help and Documentation	87
Figure 79	Display of Navigation	88
Figure 80	Context Menu for Comparing Versions	90
Figure 81	Comparison View – Comparison of two Versions	92
Figure 82	Context Menu for Completely Transferring a Version from Local Version Management	92
Figure 83	ABAP Doc Documentation of the Method	93
Figure 84	Executing a class in SAP GUI	95
Figure 85	Result of Execution	95
Figure 86	Selection of the Project	96
Figure 87	Output to the Console	97
Figure 88	Start the Data Preview from the Table	98
Figure 89	Display of the Data Preview	98
Figure 90	Navigation via Associations	99
Figure 91	SQL Console	99
Figure 92	Object Types of the CDS in Navigation1	00
Figure 93	Different Files Define the Properties of a CDS View Entity1	01
Figure 94	Overview of a CDS View Entity Using Element Info1	02
Figure 95	Calling the Dependency Analyzer via the Context Menu1	03



Figure 96 SQL Dependency Tree
Figure 97 SQL Dependency Graph 104
Figure 98 Complexity Metrics 105
Figure 99 Active Annotations of a View
Figure 100 Debugging Perspective in Eclipse
Figure 101 Values of the Variables in the Debugging Perspective
Figure 102 Creation of a Log Point via the Context Menu
Figure 103 Attributes when Creating a Log Point 112
Figure 104 Log Points View in the Debugging Perspective
Figure 105 ABAP Trace Requests in the Debugging Perspective
Figure 106 Context Menu of a Trace
Figure 107 Overview of the Properties of a Trace
Figure 108 Aggregated Overview of a Trace History
Figure 109 Jump to the SQL Trace
Figure 110 Getting started with managing User Parameters
Figure 111 Setting the User Parameter HDB_OPEN_STUDIO 117
Figure 112 Selection of eclipse.exe as a New Way to Display *.plv Files 118
Figure 113 Selection of a Specific Selection
Figure 114 Download the *.plv File
Figure 115 Query Execution Plan Display 119
Figure 116 Creation of ABAP Cross Traces
Figure 117 Detailed View of Operations 122
Figure 118 Subscribing to Popular RSS Feeds
Figure 119 Multiple Runtime Errors within a Feed 124
Figure 120 View of an SAP Gateway Error from the Error Log 125
Figure 121 Getting Started with the Eclipse Marketplace
Figure 122 Sample Search for Plug-ins in the Eclipse Marketplace
Figure 123 Installation of New Software via the Context Menu 133
Figure 124 Entering the Update Site
Figure 125 Switching between Indices
Figure 126 Adding and Setting Properties141
Figure 127 Components of a "minimal" ADT Installation
Figure 128 Components of a Minimal ADT Installation



Figure	129	Components of the SAP Update Site 1	46
Figure	130	Ability to View the Settings that have Already Been Saved1	47
Figure	131	Settings that Have Already Been Saved1	47
Figure	132	Switch to "Advanced Mode" 1	49
Figure	133	Switching between Indices 1	50
Figure	134	Selection of the Product Version1	51
Figure	135	Selection of the Stream1	52
Figure	136	Querying other Variables1	53
Figure	137	Getting Started with Global Settings1	57
Figure	138	Setting for the Dark Theme1	58
Figure	139	Source Code Indentation Setting 1	58
Figure	140	Example of Setting1	59
Figure	141	Result Image in Source Code1	59
Figure	142	Color Settings to Highlight the Keywords in the Source Code1	61
Figure	143	Administration of ABAP Templates in Settings1	62
Figure	144	Inserting the Template into the Source Code1	62
Figure	145	Getting Started with Project-specific Settings1	64
Figure	146	Possible Settings for the Pretty Printer / ABAP Formatter1	65
Figure	147	Move the View via the Label/Tab1	65
Figure	148	The Markings Indicate the Placeability of the Window1	66
Figure	149	Display of Stacked Views1	66
Figure	150	Minimizing View Groups1	66
Figure	151	Restoring the View Groups1	67
Figure	152	Closing a View1	67
Figure	153	Displaying a View1	68
Figure	154	Switching Between Different Perspectives1	68
Figure	155	Reset a Perspective1	69
Figure	156	Save a Perspective1	69
Figure	157	Naming the new Perspective1	70
Figure	158	New Perspective with Name1	70
Figure	159	Possible Setting of the ABAP Perspective1	71
Figure	160	Possible Setting of the Debugger Perspective1	71
Figure	161	Dialog for Adjusting the Sort Order of Open Editors1	73

Table of Figures

Figure	162	Examples of Available Operations in the Context Menu 174
Figure	163	Clipboard History (Keyboard Shortcut Ctrl+Shift+V) 175
Figure	164	ABAP Favorites View
Figure	165	Context Menu of a Folder in ABAP Favorites View 178
Figure	166	Colored Highlighting of the Status Bar per Project + Test Status 179
Figure	167	Management of Packages for which Unit Tests and/or ATC Test Runs are Scheduled
Figure	168	View for Managing the Stored Access Aata of ABAP Systems
Figure	169	Context Menu on Project in Project Explorer
Figure	170	Status Bar in the Eclipse Window
Figure	171	Classic Outline View
Figure	172	ABAP Code Before Quick-Fix Execution
Figure	173	ABAP Code After Quick-Fix Execution
Figure	174	Example of Quick Fix Availability on a SELECT Statement
Figure	175	SELECT Statement after Applying the Quick Fix 185
Figure	176	View "Tag Manager"
Figure	177	Search-Dialog with "ABAP Object Search"
Figure	178	Search Dialog on Page "ABAP Object Search"
Figure	179	View "CDS Analyzer" - Top-Down-Analysis 189
Figure	180	Search-Dialog with "ABAP Code Search" 190
Figure	181	Eclipse Bundle "Eclipse IDE for RCP and RAP Developers" 192
Figure	182	Plug-in Project Wizard – Entrance 195
Figure	183	Plug-in Project Wizard – Content
Figure	184	Plug-in-Projekt in the Project Explorer View
Figure	185	Feature Project Wizard - Entrance 198
Figure	186	Feature Project Wizard - Plug-in Selection
Figure	187	Update Site Wizard 201
Figure	188	Compiler Settings in the Eclipse Settings Dialog
Figure	189	Dialog for Adding an Update Site
Figure	190	GitHub Repository Settings for GitHub Pages
Figure	191	Example of a Simple Transformation for the Transformation of ABAP <-> XML
Figure	192	Sample EMF Model for Serializing XML <-> Java Object



Index of Tables

Table 1 Comparison of Different Installation Options	. 136
Table 2 Terms in Oomph	. 138
Table 3 Key Elements	. 142
Table 4 Most Important Elements of a Project	. 143
Table 5 Features of Eclipse Ini Task	. 148
Table 6 Authors	213



Introduction

Why a DSAG guide on ABAP Development Tools (ADT) in Eclipse?

Maybe you're like some of us and you're just getting started with Eclipse? Or you are already a professional and already very familiar with ADT but need basic information and a handout to make this development environment palatable to your colleagues. Or, you belong to the group of regular Eclipse users for a programming language other than ABAP and would like to know what works differently in ABAP than in JAVA, for example.

From a lot of feedback at AK Development events, it quickly became clear that a guideline created by DSAG members in the "tradition" of earlier guidelines (e.g. programming guidelines or ABAP Test Cockpit) would meet with great interest. Fortunately, in response to the AK Development call in February 2022 by Sebastian Freilinger-Huber, many volunteers came forward to create a supporting handbook, so that the work could be started while spring.

A look at the chapter list shows that we have tried to cover as many aspects of development with ADT as possible and that the individual chapters are also aimed at different target groups. Pick the topics that are most interesting to you! In appropriate places, we refer to further sources so as not to overload this guide, but still give you the opportunity to easily access detailed and further information.

We hope that you will find the information which is helpful to you in the guide, regardless of your previous experience with Eclipse and ADT, and hope you enjoy reading the following chapters!

Motivation for and topicality of the guide

Knowledge advantage, influence, and network – these are the three most important aspects from the point of view of the German-speaking SAP User Group e.V. (DSAG).

This guideline was initiated by members of the DSAG Development Working Group and addresses the first aspect: knowledge advantage for users and partners.

In recent years, extensive knowledge and experience has been compiled in numerous DSAG member companies about the framework of ADT with Eclipse. Our goal is to make these findings available to the general public in an appealing form.

Since ADT are continuously developed by SAP (and also via plug-ins from the community), this guide cannot represent a completed state. Rather, it should be understood as a "living document". New findings are constantly integrated, and knowledge already gained is updated. For this reason, in addition to a published



version 1.0 (PDF), a continuously updated Git repository is provided for managing the content. You can use the already established mechanisms of Git (Issues) to place your feedback on the guide. The current version of the guide is continuously available via the website, the PDF version is updated at regular intervals. However, due to the effort to maintain the repository in a continuous way, the feedback mechanism is only supported in German language.



1 Introduction Eclipse and ADT

1.1 Eclipse ... Why?

Why is Eclipse the right platform for ADT and not VSCode, NetBeans or any other development environment? It will not be possible to give a fully comprehensive answer here. The fact is, however, that SAP has strategically decided to use Eclipse as the basis for the future ABAP development environment (as the successor to the classic ABAP Workbench) and plans to continue to pursue this approach. This can also be seen in the roadmaps of the ABAP platform and can be read there.

However, it must be mentioned that the connection to Eclipse has created an open interface (between the development environment and the SAP backend) that can be used by other environments. For VSCode, for example, there are already extensions based on these interfaces. However, the completeness and usability are not discussed in the guide. If you need more information about this, you will quickly find suitable results via the usual search engines.

1.2 A brief look at the history of its creation

The origin of Eclipse is at IBM and became independent in 2004 with the Eclipse Foundation. Eclipse is implemented in the Java programming language and is openly available under an open-source license.

Initially, the new Eclipse versions were named after objects from space (including Jupiter's moons). For the sake of simplicity, however, in 2018, the Eclipse Foundation opted for a comprehensible naming convention for the individual versions: < (year)-(month) >. At the same time, the company switched to significantly faster (three-month) release cycles.

1.3 The concept of extensions

The basic installation of Eclipse is already delivered with a basic set of functions and extensions. The scope of these functions and extensions depends largely on which version you select for download on the website. From a technical point of view, the core of the application is always identical. All additional functionalities are delivered by means of extensions – this also applies to the ABAP development tools. This means that you can always add or remove all extensions afterwards from your installation.

Typically, you obtain the extensions via the Eclipse Marketplace, which you can also access directly from the tool.



1.4 Difference between perspective and view

An important feature of Eclipse is its customizability. This will be discussed in detail in the following chapters. It is essential to understand the basic terminology.

A view is an independent part of the tool that is either already included in the basic installation or was added later via the Eclipse extension. This view can be arranged by the user at different locations within the development environment.

A perspective contains a specific layout of Eclipse, i.e., all displayed views and their position. Perspectives thus give you the focus for an activity with all the views you need for it. For details, see <u>Chapter 3 - Working with ADT</u>.

Knowledge of the terminology is essential to be able to work efficiently with Eclipse. Documentation of all kinds uses this terminology to explain functions. This also applies to this ADT guide.

1.5 The power of keyboard shortcuts

A distinctive feature of Eclipse is the use of keyboard shortcuts. In the beginning, you will spend a lot of time trying to find and memorize the functionalities you are used to from the classic ABAP Workbench (transaction: SE80). In contrast to SE80, however, Eclipse is optimized to be operated with keyboard shortcuts. Keep that in mind and take advantage of the opportunities to internalize the keyboard shortcuts. For more information, see Chapter 3 - Working with ADT.

1.6 Which Eclipse feature is available in which version?

Since the functionality of the ABAP Development Tools is continuously being expanded by SAP, it is important to check which functions are available in the ADT version you are using. It should be noted that some of the functions provided also depend on the release of the SAP system used. To find out the current status of the latest updates of ADT, the official sources of SAP are the means of choice, as they are continuously maintained.

For Cloud:

https://help.sap.com/docs/BTP/5371047f1273405bb46725a417f95433/ab03dcd9072 f4a2d85c945d05929d3fb.html



For On-Premise:

https://help.sap.com/doc/2e9cf4a457d84c7a81f33d8c3fdd9694/Cloud/en-US/inst_guide_abap_development_tools.pdf



2 Motivation for ADT

In this chapter, we will introduce you to several reasons for using the ABAP Development Tools (ADT) as a development environment. At the end, we also go into what is sometimes the biggest "sticking point" for long-time ABAP developers. We give hints, why the aspects, which are often considered as disadvantages, are actually advantages.

We would be pleased if we can convince you of the benefits of ADT with this chapter or at least arouse your curiosity to study this guide.

2.1 You want to apply and implement modern ABAP.

The first and most obvious reason for a comprehensive use of the ABAP development tools in Eclipse is the strategic orientation on the part of SAP. The ABAP Development Tools were introduced in 2012 and have developed into a stable development environment with a wide range of functions over these ten years.

In contrast, the SAP GUI-bound development tools with their most prominent example, the ABAP Workbench, are in maintenance mode. This means that bug fixes will continue to take place here, but no new features will be delivered.

This may not be the most motivating reason in the short term, but it will one day catch up with every developer in the SAP environment. Therefore, we recommend that you start and switch to ABAP development tools today rather than tomorrow.

For this reason, SAP recommends the use of ADT as the standard environment for ABAP development to benefit from new functions and corrections with each release. The actual range of functions depends on the release status of the ABAP stack of the SAP systems used. A rough overview and further information can be found in the section: <u>Chapter 1 - Introduction to Eclipse and ADT</u>.

2.2 You want to use one development environment for everything.

2.2.1 Technological aspect

With the ABAP Development Tools (ADT), you can develop not only for on-premise systems such as SAP ERP or S/4HANA. ADT are also the only way to perform developments for cloud systems such as the Business Technology Platform (BTP).

This circumstance is particularly important for ABAP developers who develop in the SAP Cloud ABAP Environment ("Steampunk"). In this context, classic, SAP GUI-



oriented development tools cannot be used, and developments can only be carried out with ADT.

Closely related to this is also the work with some new development artifacts. The creation or maintenance of CDS views is only possible with ADT. And SAP's new programming model, the "RESTful Application Programming Model" (RAP), can only be used with ADT.

2.2.2 Development on multiple system lines

One development environment for everything applies to ADT but also if you are allowed or must work on several development systems. In the Eclipse environment, various systems are listed as projects, and you can access them very conveniently and clearly. By means of the working sets (see chapter 3 - "Working with ADT"), the systems can be grouped into umbrella terms and even color-coded using additional plug-ins (see chapter 7). There is no need to open the systems from the SAP Logon Pad, log on to the system and open the SE80. This results in numerous other synergies in the work, which are described in this guide.

2.3 You want to break free from the technical limitations of the SAP GUI.

2.3.1 Six sessions in one system are simply not enough

SAP GUI-based development tools such as the ABAP Workbench are subject to various limitations due to their execution in the SAP GUI. These are not relevant when you work with ADT.

This includes, for example, the dependence on the maximum number of simultaneous SAP GUI sessions (modes, cf. parameter "rdisp/max_alt_modes"). This value is specified individually for each system called via SAP GUI. By default, there are at most six simultaneous SAP GUI sessions per user in the same system. This restriction does not apply to ADT.

If you are a developer working with different systems at the same time, you benefit from another advantage: Each connection to a system is maintained as an "ABAP Project" for on-premise systems or "ABAP Cloud Project" for cloud systems within ADT. The connections to these systems can be used simultaneously. This allows you to copy source code from a source system to the clipboard and paste it into a target system that is open at the same time, all within the same Eclipse application window.



In addition, a simple code comparison between different systems, even between systems without an existing RFC connection, is also possible in this way. Displaying multiple SAP GUI windows as needed previously is no longer necessary.

2.3.2 You don't want to lose code due to network interruptions.

Due to the technical conditions of the SAP GUI, it is necessary that the network connection is stable and uninterrupted. If there are network interruptions while you are writing code in a GUI window, it may happen that the work of the last few minutes was in vain because SAP GUI has lost the connection to the server and closes the window.

With ADT network interruption is no longer a problem. The Eclipse window remains open even if the connection is lost. Once the connection is restored, the code can be saved in the SAP system.

If major problems occur in the network, the code can simply be cached in its entirety as text using copy-and-paste in an alternative text editor until the SAP system is available again.

2.4 You like to improve your ABAP code through refactoring.

The maintainability of development artifacts is a central challenge of software development. To achieve good maintainability, the development procedure is often based on clean code principles (see Clean ABAP). To comply with the clean code principles, repeated revision of development artifacts, also known as refactoring, is essential.

The ADT support typical refactoring tasks with the help of the Quick Assists functions, which also include the Quick Fixes. The Quick Assists are context sensitive. For example, you can outsource complex source code sections to smaller and independent methods, which significantly increases comprehensibility and thus maintainability. Without the support of Quick Assists, such an iterative approach is much more labor-intensive and error prone.

Further and more detailed information on refactoring and the tools provided to you in ADT for this purpose can be found in Chapter 3 - Working with ADT.



2.5 You like to adapt the look and feel of your development environment to your wishes.

ADT are based on Eclipse, a widely used open-source development environment. Strengths of this development environment are, among other things, the wide range of customization options such as in the display (cf. Views and Perspectives in Chapter 3 - Working with ADT) or in the shortcuts (keyboardshortcuts) and the support of helpful functions such as e.g. a comprehensive search.

As a user of Eclipse, you can therefore customize and use the development environment in a variety of ways, which allows each user to better support their preferred way of working.

Since ADT allow access to different SAP systems at the same time, you can work consistently across all systems. This is in contrast to working with SAP GUI-based development tools such as the ABAP Workbench, which must be individualized separately in each system.

With the so-called workspaces, you have the option of saving different configurations of the working environment. For example, different projects, different favorite packages, which views and objects are open and even the place in the code where the last work was done are stored in the respective working set. In addition, different workspaces can be opened in several Eclipse instances running in parallel. This gives you the opportunity to put together the most efficient environment depending on the project, customer, or task. See Chapter 3 - Working with ADT and Chapter 6 - Best Practices Eclipse Configuration for information on how to use workspaces.

2.6 Would you like even more tool-based support?

If the range of functions provided by SAP is not sufficient for you, you can extend it with additional plug-ins that are available as a supplement to ADT.

A selection of these plug-ins is presented in Chapter 7 - Plug-ins . With the right knowhow, you can also create your own plug-ins and make them available to the SAP community.

In the past, various ABAP developers have already made use of this option and developed their own plug-ins, which they make available to the SAP community free of charge. An example of this is the ABAP Quick Fix. This is an extension of the quick fixes offered in the standard ADT to support the automatic conversion of the classic ABAP syntax into the respective modern ABAP syntax counterpart.



2.7 You already know Eclipse from other programming languages? Great!

Eclipse as the basis of ADT is already in use in various companies. This is because it is one of the most widely used development environments for JAVA, for example. JAVA, on the other hand, is a frequently used programming language (cf. TIOBE Index) and therefore plays a role in many development projects. In addition, Eclipse can also be used for other development purposes, e.g. for development in the Python programming language or working with data in the Extensible Markup Language (XML) format.

So, there is a certain probability that developers and companies already have previous knowledge of using Eclipse as a development environment. This makes it easier and faster to extend the use of ADT than to use a completely new, possibly completely unknown development tool.

2.8 You are interested in new things.

Finally, we would like to point out the interest of many people in new things and the associated attractiveness as a motivating factor for the change. If the reasons for switching in the previous sections were more factual aspects, this one is about emotions and values deeply rooted in people: the interest in something new and the attractiveness that something new has. Or in short: about progress and change.

For some people, the standstill with occasional small changes means a certain security in everyday working life. However, this security is deceptive, as the (IT) world is changing rapidly, and existing technologies and methods do not always offer the right answers to current and future challenges.

Especially in the ABAP world, a lot has changed in the last ten years. After numerous new language constructs have enriched ABAP, new artifacts such as CDS views and new programming models such as the ABAP RESTful Application Programming Model (RAP) have been added. And technological change such as the path to the cloud cannot be dismissed out of hand. Consequently, this also requires the further development of the tools used in development.

In addition, a technological standstill can also lead to the exclusion of subsequent generations of developers, as they often look at technologies with a completely different perspective, compare their previous experience with other development environments and programming languages and make their evaluation on this basis.



2.9 The benefits for the organization

With this guide, we want to motivate not only the ABAP developer to use ADT. There are also numerous advantages for the organization through the extensive use of ADT in the SAP development area.

In addition to the higher efficiency of the developers, among other things through better tool support for creating and improving the code, future viability and uniform tooling are again the most important reasons for the organization to promote and demand the use of ADT in the SAP development teams.

To enable smooth use for the individual developer and to avoid obstacles to a changeover right from the start, and thus to achieve a broad use of ADT, the following topics must be clarified centrally and made available to developers in the form of generally available documentation:

- Framework conditions and general information on ADT
- Installing Eclipse
- Access from Eclipse to resources on the Internet (updates and plug-in installations)
- Developer permissions for ADT objects in the backend (S_RFC for ADT*)

Again, this is an initial investment, but it pays off as soon as the development teams work uniformly with ADT and the advantages of the development environment can be fully exploited in their daily work.

You can find a lot of information, assistance, and best practices in Chapter 5 - Installation, Distribution and Update Strategies.

2.10 You are missing the form-based editor

ADT use a text-oriented representation of development artifacts in many places, which you are familiar with from other development environments such as Microsoft Visual Studio Code (VSCode). This eliminates the previous, form-based presentation that you are familiar with from the SAP GUI-oriented development tools.

This is certainly one of the biggest hurdles for many ABAP developers who want or should switch from GUI-based tools to Eclipse. To overcome this hurdle, you will find a detailed introduction to working with ADT in Chapter 3 - Working with ADT.

This change in presentation is also accompanied by a certain change in the way of working. There is no need to frequently switch between different SAP GUI interfaces that are in a specific context. Instead, there is often "only" text - i.e., instructions that



are entered or read. This leads to a strong focus on the actual instructions, their effects, and relationships (syntax and semantics).

An example of this changed representation is the signature of a function module. With the SAP GUI-based development tools, the signature of a function module consisting of IMPORT, EXPORT, CHANGING, TABLES and EXCEPTIONS is displayed as five separate registers. In ADT, the signature is displayed and maintained as text. There is no need to switch between different context-bound registers.

After the initial adjustment, you will certainly quickly recognize the advantages of the text-oriented way of working, which result from the omission of the navigation steps through the GUI and from the numerous supports such as code completion and quick fixes.

2.11 Why ABAP Development Tools

If you have not yet been convinced by the advantages of ADT described so far, we would like to motivate you to put yourself in the shoes of ADT users with quotes from the authors of the guide. Perhaps you will find the impetus here to take up the topic after all.

Michael Keller: "Clean ABAP without ADT and thus the support of the quick fixes is unthinkable for me - after all, they save the developer a lot of time and work."

Florian Henninger: "Refactoring without ADT is a bit like trying to eat soup with a fork - it can work, but no one does."

Bärbel Winkler: "While helping to create this guide, I learned many good reasons to work with ADT more often from now on than I did before."

Jens Zähringer: "Although the ABAP development tools have been available for over 10 years, I recently made the switch from ABAP Workbench to ADT for myself. The transition wasn't without its challenges, but in the end, it was definitely worth it!"

Peter Luz: "Using the where-used list, you can quickly determine where a method is used and how it is called there. Then rename this method that is used multiple times. Then extract a piece of code from it into your own method. Finally, compare the code on the central development system with the version on the Q machine of a system line. In ADT, by using a few keyboard shortcuts it is done in seconds. This makes creating and revising software fun and helps to significantly improve the quality of the software by using the tools described here in the guide. For me, creating ABAP software without ADT is now unthinkable."

Michael Biber: "Yes, SE80 is now quite good. However, I see this like the switch to object orientation: At the beginning you ask yourself 'Why?' and see existing stumbling



blocks. However, once you have experienced the other side (object orientation, ADT ...), you don't want to go back. For me, the advantages of better clarity (element info), theoretically infinitely open sources in parallel, live syntax checking and many more outweigh all the small media breaks (\rightarrow older SAP releases) and other approaches."

Björn Schulz: "Without ADT, I would be much slower to get information out of the system and probably wouldn't be able to cope with only six modes."

Dr. Wolfgang Röckelein: "Is there ABAP development without ADT?" "No clean code without refactoring, no refactoring without ADT."

Uwe Fetzer: "There is no faster, more convenient, and safer way to develop ABAP Clean Code."

Sebastian Freilinger-Huber: "It's better to leave the 'SE80 comfort zone' today than tomorrow - it's worth it. If you still have doubts, you will find numerous arguments for the switch in the following chapters."



3 Working with ADT

The ABAP development tools offer many functions, which also offer correspondingly numerous possibilities for use. This can be confusing for developers who are new to ADT.

The first step is always the hardest. At the beginning of this chapter, we would like to make it easier for beginners to get started in the ADT world using the example of creating a class. The procedure shown here can then be transferred to other development objects, and the entry is done.

The other functionalities are described in the following section, and hints are given, and best practices are explained on how the numerous tools and aids can be used in daily work and offer added value.

In addition to ADT beginners, developers experienced in ADT will also find many useful hints and perhaps also new things for their daily work with the ABAP development tools.

With the SAP ABAP Development User Guide, hereinafter referred to as the User Guide, SAP provides the official documentation for ADT. For further information on individual functions, the links to the corresponding section in the user guide are noted.

3.1 Introduction: Basics of working with ADT

3.1.1 Getting Started with ABAP Development Tools

This section is intended for developers who have not yet worked with ADT and want to get started. In the user guide, in the "Getting Started" section, there is a section, where all functions are explained in detail. In this guide, we would like to make it as easy as possible to get started by means of a step-by-step description of how to create an ABAP class and show the advantages of using the ABAP development tools.

The following steps are described step by step:

- Setting up the project
- Setting up the Favorite Packages
- Creating a class and a method
- Working on code and refactoring

For the purpose of general traceability, the examples shown here are displayed in an instance of the BTP Trial Account, but they can be used in common on-premise systems without any problems.



Developers who have already gotten started and want to get an overview of the individual functions can skip this section.

3.1.2 The shift from form-based to text-oriented code creation

In contrast to SAP GUI-based transactions such as SE80 or SE24, ADT does not have a form-based editor. The creation of classes (and also function modules, etc.) is purely text-based in ADT. For development tools that are not yet available in Eclipse depending on ADT and backend version, these transactions can be accessed from Eclipse in an integrated manner.

It also takes some time to get used to swapping the **F2** and **F3** keys. While the F3 key is used as the back key in the SAP GUI, the F3 key in ADT is used for forward navigation, the F2 key for context-sensitive help.

This is sometimes the biggest hurdle for getting started, as long-established practice and familiar work processes change with the switch to ADT. And if you want to go fast, you like to fall back on familiar and well-known ways of working.

The introduction and the changeover therefore require time and effort of practice. But the initial additional effort pays off after a short time. Because after a little acclimatization and practice, the points described are no longer a problem. The numerous functions offered by the ABAP development tools make it easier to write and revise ABAP code and thus increase the efficiency of development. Therefore, the switch from SE80 & Co. on ADT in Eclipse can be seen as a personal investment in an efficient and future-proof way of working.

3.1.3 Context of the exercise

The example shown here is deliberately simple, as the basic functions and working methods of ADT are primarily to be presented.

We would like to create a small class that offers the following functions:

- 1. Determination of flights from the table /DMO/FLIGHT as entered.
- 2. Calculation of the available seats of the flight.
- 3. Calculation of the fare based on an additional percentage fee.

The class serves as an internal service class and does not provide a UI or output data.

It shows the basic working methods and most frequently used functions that lead to efficiency gains during code creation and modification.



3.1.4 Connecting the Development System – New Project

A development system is represented in ADT in the form of a project. Therefore, to link a development system to ADT, we need to create a new project.

In a new installation of ADT, a new project is created via

 $\mathsf{File} \to \mathsf{New} \to \mathsf{ABAP} \ \mathsf{Project}$

0	WS-202	22-06 - Eclip	ose IDE					
File	Edit	Navigate	Search	Project	Run	Window	Help	
	New				AI	t+Shift+N⇒	· 🖻	ABAP Project 🔗 🔻 🖢 🖛 🖏 🖛 🏷 ⊂≯ 🛟
	Open	File					Ŕ	ABAP Cloud Project
	Open	Projects fro	om File Sy	stem				Project
	Recen	nt Files				;	0	ABAP Class
	Close	Editor				Ctrl+W	0	ABAP Interface
	Close	All Editors			Ctr	l+Shift+W	轚	ABAP Package
	Save					Ctrl+S		Example
	Save A	4s			_			Other Ctrl+N
	Save A	All			Ct	rl+Shift+S	Г	
	Reven	t						
	Move							
Z	Renar	ne				F2		
69	Refres	sh				F5	>	
	Conve	ert Line Deli	miters To			;		
8	Print.					Ctrl+P	F	
è	Impo	rt						
4	Expor	t						
	Prope	rties				Alt+Enter		
	Switch	h Workspac	e			;		
	Restar	rt						
	Exit							
							_	

Figure 1 Creating an ABAP Project in Eclipse

When you create an ABAP project for on-premise systems, the list of systems linked in the SAP logon is displayed. The login data must be stored if no SSO is used and in the last step the project can be given a descriptive name. The language English is selected as the default.

The newly created project and the associated development system is displayed in the so-called Project Explorer.





Figure 2 The Project Explorer

The Project Explorer is the central entry point and object catalog after the corresponding development system has been opened. The objects are displayed in hierarchical form based on the packages, as already known from the SE80. In daily work, the objects to be edited are opened from this.

The main workspace is the Favorite Packages node. To add the package to the Favorite Packages into which the class is to be inserted, execute the "Add Package" command using the context menu.



Working with ADT

듣 WS_ADT_DSAG_CLOUD - E	clipse	IDE			—			×
File Edit Navigate Search	Pro	oject Run Window Help						
i 📬 👻 🗐 👘 🗐 🗮 🔯 💋	9	😼 : 🎼 🎝 🖨 🖏 : 📴 🗉	η 🔅 🕶 🚺) - 🚷 - (🌯 🕶 🛷 🕶			
월 - 월 - 10 - 10 🔶 -	<> →				Q	: 🗈	む	4
Project Explorer X	CDS	Navigator 🗖 🗖)[
v 🛃 CLOUD_ABAP-202210 [TRL, 1	00,						
🗸 📌 Favorite Packages (2	2.727)							
> 🖶 ZLOCAL (22.727)		New		>				
Eavorite Objects (0)	ß	Duplicate Tree						
> 🌣 Released Objects (3.1		Open ABAP Development Ob	ject					
	Þ	Сору		Ctrl+C				
	ж	Delete		Delete				
	₽	Add Package						
<	Ð	Refresh		F5				
🗄 Outline X	Q	Coverage As		>	💷 B) 🔝 F) 具	T		
There is no active editor that p		Coverage As		>			7	8
	0	Run As		>	~			
	*	Debug As		>				
		Profile As		>				
	>	Expand Tree by						
		Configure Tree	Α	Alt+Enter				
			<					>
ᄸ [TRL] Favorite Packages								: 💡

Figure 3 Adding Packages to Favorites

3.1.5 Creating a class in text mode

To create a new ABAP class, navigate to the desired package in the Project Explorer, right-click on the context menu, and find the command

 $\text{New} \rightarrow \text{ABAP Class}$



Working with ADT

듣 WS-2022-06 - Text	Elem	ents ZCB998PL_CL_ABAP_TEST [TRL] - active - BTP-Trial-	2022-10	· Eclipse IDE	
File Edit Navigate	Sea	rch Project Run Window Help			
🔁 - 🔛 💿	ìQ, [E 9 R 🛛 🔻 🕷 🖏 🕰 💷 🗉 🗉	*	0 • 🤮 • 💁 • 🔗 • 🖢	• @ • *;-
陷 Project Explorer 🛛	:	🖏 Relation Explorer 🛛 🗖 🗖 💻	., 1988	Land test X	
		E 🕏 🏹 🖇 🔰 1			
✓ 10 BTP-Trial-2022-1	10	L THE CROMMONICANT, INC.			
🗸 📌 Favorite Pac	kages	; (56.054)			
> 🌐 ZLOCAL	(56.05	4) Generated Package for ZLOCAL			
> # ZZ_CB998		New	>	ABAP Class	
> 😭 Favorite Obj	2	Derive New Tree		ABAP Interface	
> 💭 Keleased Ob		New Kennel des Transfer De sum ent			
	2	New Knowledge Transfer Document		авар раскаде	
		Open		Other ABAP Repository Object	
		Open in Project Ctrl+4	Alt+P >		
		Open ABAP Development Object			
		Shawla Alt St	6. M.S		
		Show in Alt+Shi	rt+vv >		
	Þ	Сору С	trl+C		
	×	Delete D	elete		

Figure 4 Creating a New ABAP Class in Project Explorer

А	window	opens	in	which	the	data	of	the	class	can/	'must	be	specified
€	New ABAP Clas	55								—		×	
AB/ Cr	AP Class eate an ABAP cl	ass									C		
Pro	ject: *	BTP-Trial-20	022-10								Browse.		
Pac	:kage: *	ZZ_CB99		47. Bank	-						Browse.		
	Add to favorite	packages											
Nar	me: *	Z_FLIGHT_I	NFO										
Des	cription: *	Information	n abou	t free capacit	ty and pri	ices							
Ori	ginal Language:	EN											
Sup	erclass:										Browse.		
Inte	erfaces:										Add		
											Remove	2	
(7					< Ba	ck	Next	>	Finish		Cancel		

Figure 5 Properties Dialog: Creating an ABAP Class



If required, the super class and interfaces to be referenced can already be specified here. However, this can also be done later text-based directly in the source code. After clicking on "Next", the window for selecting or creating the transport request opens.

New ABAP Class	-							\times	
Select Transport Request For Z_FLIGHT_INFO (CLAS), the selected transport request will be used 							00	0	
Choose from requests in which I am involved						Configure Colum			
type filter text									
Transport Request	Owner CB99	Target	Descript ZZ_CB9	tion 9	TRAIN, MAR	8	C	TS Pro	
<	est							>	
CTS Project:							Browse.		
O Enter a request nur	mber						Browse,		
?			< Back	Next >	Finis	h	Cance	:1	

Figure 6 Transport Request Dialog

After clicking on "Finish", the class is created, and it can be found in the object tree of the Project Explorer and in the source code editor on the right side of ADT.



듲 WS-2022-06 - Global Class Z_FLIGHT_INFO [TRL] - inactive, locked - BTP-Trial-2022-10 - Eclipse IDE — 🛛 🛛									
File Edit Navigate Search Project Run Window Help									
😁 🕶 🐚 🖳 🔍 👅 🗣 😿 😒 🖉 🌜 🖉 🔄 🐨 🕼 🖏 🚭 🗊 🗐 👖 🔅 🍫 🗘 🗣 🖓 🕶 🖓 🕶 🖓 🕶 🖓 🕶 🖓 🕶 🖓 🕶 🖓 🕶									
			Q	: 😭	患				
			`			5			
						-			
	► G Z_FLIGHT_INFO ►								
✓ → BTP-Trial-2022-10 [TRL, 100, CB99 EN]	10 <u>CLASS</u> z_flight_info DEFINITION				^				
Favorite Packages (50.054)	3 FINAL								
> # ZLOCAL (50.055) Generated Package for 2	4 CREATE PUBLIC .								
	5								
	6 PUBLIC SECTION. 7 PROTECTED SECTION.								
Z ELIGHT INFO Information abo	8 PRIVATE SECTION.								
> 👷 Favorite Objects	9 ENDCLASS.								
> 🔅 Released Objects	10								
	12								
	13⊖ CLASS z_flight_info IMPLEMENTATION.								
	14 ENDCLASS.								
🗄 Outline × 📄 🖓 🖌 🖇 🗖 🗖	<	-			>				
C Z_FLIGHT_INFO	Sclass-relevant Local Types Class-relevant Local Types	Macros							
	🖹 Problems 🗙 🔲 Properties 🔚 Templates 💷 Bookmarks 🗟 Feed Reade	er 📮 Transport O	rganizer		- 6	3)			
					7	8			
	0 items	1 1							
	Description	Resource	Path			L			
						_			
						+			
	<	· · · · · · · · · · · · · · · · · · ·				>			
Project: BTD-Trial-2022-10	0 items selected								
	o nem serecco								
1									

Figure 7 Display of the new class in ADT

All further operations on the class are now performed in the source code, i.e., all types, data definitions and methods are written text based as source code in the class. This seems complicated and unusual at first, but ADT offer some functions that make the elaboration of the class very efficient.

3.1.6 Definition of a method in the class

Now we want to create the first method of the class, which reads data from the /DMO/FLIGHTS table and outputs the number of free seats for a defined flight.

We limit ourselves here only to the core functionalities and will not provide any additional functions for output, etc.



In ADT, an ABAP class is divided into the main areas "Definition" and "Implementation". Accordingly, for our first method, we will first define the method with its parameters in the "Definition" section and then perform the implementation with the source code. The so-called "Quick Fix" function will save us typing work.

To create a method, navigate to the "Global Class" tab in the

CLASS <classname> DEFINITION

and places the cursor in the visibility area of the class in which the method is available. In our case, the method should be visible to other users and should therefore be defined in the PUBLIC area. The definition of the method is introduced with the METHODS keyword.



Figure 8 Editing the Class

Already at this point, one of the main advantages of ADT over the SE80 becomes apparent. As soon as code has been entered, the syntax check runs automatically and indicates whether the code is syntactically correct.



Since the completion point does not exist, ADT displays a hint (as a hover message) in the left bar as well as the syntax error in the view problem. There is no need for an extra syntax check. Just a small thing that means a significant increase in efficiency in daily work. At the latest when you make a change in the GUI-based tools for various reasons, you will miss this function.

3.1.7 Automatic completion and formatting of the code

We complete the method definition with the creation of the parameters and the completion point. The use of code completion makes our work much easier. To do this, we only enter the first two to three letters of the desired keyword. The keyboard shortcut **CTRL+SPACEBAR** shows us the appropriate keywords. These can then be selected from the suggestion list using **TAB+ARROWKEYS** For further automation of code creation, the use of templates is recommended (see section Code Template).



Figure 9 Example Code Completion for the Import Parameter

After the import and return parameters have been defined, the cursor is positioned in the text area of the code and the formatting of the code is executed using the right-click → Source Code → Format context menu or the keyboard shortcut Shift+F1.



This corresponds to the Pretty Printer in the GUI transactions. The code is then saved using

CTRL+S.

If there is a syntax error, the problem view and the color marking in the code will show the error. If the code is syntactically correct, the artifact can then be activated using CTRL+F3.

After practicing the new, mainly keyboard-based, way of working, an automatism is created after a short time, which, after writing a few lines of code and subsequent sequence of the above-mentioned key combinations, results in a formatted, checked, and saved code. This saves time-consuming surprises in the form of syntax errors when activating, as can happen in the GUI tools.

3.1.8 Implementation of the method using Quick Fix

The method is now defined, due to the missing implementation, ADT displays the following error in the problems view:

"Implementation missing for method "CALC_FREE_SEATS_FOR_FLIGHT".

This "problem" can be solved very efficiently with the help of the Quick Fixes.

To implement the method, use the Quick-Fix-Function, which can be called up via the context menu or the key combination **CTRL+1**.


듲 WS-2022-06 - Global Class Z_FLIGHT_INFO [TRL] - inactive, Id	:ked - BTP-Trial-2022-10 - Eclipse IDE	- 🗆 X
File Edit Source Code Navigate Search Project Run	Vindow Help	
i 🖻 🗝 🗐 🐚 i 🖬 i 🐹 i 🙇 💁 🔂 i 🥥 🕷 🗰 🏞 🙈 🌢	🔁 😰 🔏 🔃 🗐 🖷 👘 🗄 🛪 🖕 🖓 🖛 🚱 🖛 🚱 🖛 🍕 🖛 🏷	
		〇 : 🔂 🗼 🌠
	~	<u> </u>
Project Explorer × 🥰 Relation Explorer	<pre> «"*[TRL] Z_FLIGHT_INFO × </pre>	
🖻 🕏 🏹 🕴	G Z_FLIGHT_INFO ▶ ● CALC_FREE_SEATS_FOR_FLIGHT	
✓	10 <u>CLASS</u> z_flight_info DEFINITION	^ <u>=</u>
V 🖓 Favorite Packages (57.833)	2 PUBLIC 3 FINAL	
ZLOCAL (57.833) Generated Package for ZLOCAL	4 CREATE PUBLIC .	
V # ZZ_CB998 DSAG AL	5	
Classes (1)	6 PUBLIC SECTION.	
Classes (1) <pclasses (1)<="" p=""> <pclasses (1)<="" p=""> <pclasses (1)<="" p=""> <pclasses (1)<="" p<="" td=""><td>8 IMPORTIN</td><td>Creates an e</td></pclasses></pclasses></pclasses></pclasses>	8 IMPORTIN	Creates an e
CALC FREE SEATS FOR FLIGHT	9 (W Add ABAP Doc	in class z_fl
Z_FLIGHT_INFO Text Elements	10 Add implementation for calc_free_seats_for_flight	
> 😭 Favorite Objects	Make calc_free_seats_for_flight private PROTECTED_SE	
> 🔅 Released Objects	13 PRIVATE SECT	
	14 ENDCLASS, A Delete Calc_free_seats_for_flight	
	15	
	17	
	180 CLASS z_flight	
	19 ENDCLASS.	
	A Science Circ	-
🗄 Outline X 📄 🛱 🖉 🗖 🗍	Problems 🗙 🔲 Press 'Ctrl+Shift+1' to show ir	n Quick Assist View
✓		7 8
CALC_FREE_SEATS_FOR_FLIGHT	1 error, 0 warnings, 0 others	Vo
	Description	Resource Path
	Reference (1 item)	
	¢	
Project: BTP-Trial-2022-10	Writable Smart Inser	t 7:14:104
0		

Figure 10 Using the Quick Fix for Method Implementation

You select "Add Implementation for ..." and confirm with ENTER. To run the quick-fix commands, the code should be saved and error-free, as some quick fixes (automatic code corrections/refactorings) require this.

The method implementation here means that the section

METHOD

ENDMETHOD

in the area of the "Class Implementation" is created by ADT and thus an empty method implementation exists. The development of the method logic within this area is then your task as a developer.

You can jump back and forth between definition and implementation simply by pressing the $\mathbf{F3}$ -key.



By placing the cursor on the method name and pressing **F2** you can display the parameters of the methods. This makes it easier to write the code and saves cumbersome navigation into the definition area. Alternatively, the ABAP Element Info is available for this purpose.

In our tutorial, we use the import parameters to read the desired record using a SELECT command. Subsequently, the number of free seats is calculated and returned to the user as a returning parameter.

Again, using Code Completion, which is invoked with the keyboard shortcut **CTRL+SPACEBAR**, helps to create the code efficiently and free of typos.

To avoid writing the whole returning parameter "r_f_free_seats" and to get type information, we write the structure and the component separator "-" and use the key combination **CTRL+SPACEBAR** to display the components, which can then be selected and inserted into the code.



Figure 11 Component Selection Using Code Completion



These options for forward navigation and inline display of further element information are not available in this form in SE80 and are a further building block for increasing developer efficiency in ADT.

We have thus defined and implemented the class with the first method in ADT.

3.1.9 Renaming of Parameters – Refactoring

The ongoing optimization of existing code is an important task of every developer. This is supported by ADT in the best possible way. The different ways to refactor are explained in detail in the section: Refactoring Code with ADT. Further information can also be found in the User Guide.

We would like to present the general procedure of refactoring in detail by means of a renaming.

As usual, we used the Hungarian notation and now want to rename the parameters of the method to remove the prefixes (see ABAP Clean Code - https://github.com/SAP/styleguides/blob/main/clean-abap/CleanABAP.md).

While such an action can be very time-consuming and error-prone in the SAP GUI environment, the rename function in ADT offers a very convenient way to rename variables, parameters, and method names across all uses. This means that in all development objects that call the method, the parameters are automatically renamed. There is no need for a time-consuming search for users via the where-used list. Of course, this only works if no dynamic method calls are used. ADT cannot detect such cases.

Thus, code cleaning and refactoring with ADT can be carried out very efficiently, and the risk of resulting errors is significantly reduced compared to the manual method.

To rename the parameters, the Rename function from the context menu via right mouse button \rightarrow source code \rightarrow Rename or the keyboard shortcut

ALT+SHIFT+R or via Quickfix selection via

CTRL+1

executed.



How S-2022-06 - Global Class Z_FLIGHT_INFO [TRL] - inactive, locked - BTP-Tria	-2022-10 - Eclipse IDE							- 0	×
File Edit Source Code Navigate Search Project Run Window Help									
i 🗂 🕶 🔚 🕼 i 🖳 i 🔌 i 🚝 🗣 📈 i 🤡 🛠 🕷 🗞 🖴 😰 🕖	📴 🔲 👖 💠 🕶 🔾 🕶 🤮 🕶 💁	1	• 🔄 • 🖓 • 🏷 🗘 🔶 •	· 📑				Q 🗄 🔡	影 🔼
陷 Project Explorer 🗙 🧟 Relation Explorer 🛛 🕒 🤹 🍞 🖇 📟 🗖	C *(TRL) Z_FLIGHT_INFO ×								
✓ 5 BTP-Trial-2022-10 [TRL, 100, CB996 EN]	▶ G Z_FLIGHT_INFO ▶ ● CALC_FRI	E_SEAT	S_FOR_FLIGHT						
 	<pre>1@CLASS =_flight_info DEFIN 2 PHUL 3 FTWAL 4 CREATE PUBLIC . 5 PUBLIC SECTION. 7 METHODS calls free_sea 8 IMPORTING if_carri 9 i_f_come 10 i_f_fligh 11 RETURING VALUE(_f 12 PROTECTED SECTION. 13 PRIVATE SECTION. 14 ENCLASS . 15 16 10 10 10 10 10 10 10 10 10 10 10 10 10</pre>	ITION ts_For er_id ction_ t_date _free_ MENTAT	-flight TYPE /dmo/Carrier_id id TYPE /dmo/Connection_it TYPE /dmo/flight_date seats) TYPE 1.	4					
	19 20⊖ METHOD calc_free_seats_	for_f1	light.						
	21 22 SELECT SINGLE FROM /d	mo/fli	ight FIELDS seats max, seats o	ccupied					
	23 WHERE carrier_id	- []	Undo Typing	Ctrl+Z					
	25 AND flight_date	- 6 -	Revert File						
	26 INTO @DATA(flight 27⊖ IF sy-subrc = 0.	- ^{ca} 🖫	Save	Ctrl+S					
	28 r_f_free_seats = f1	igh 👷	ABAP Type Hierarchy	F4	-seats_occu	pied .			
	30		Quick Type Hierarchy	Ctrl+T					
	31 ENDMETHOD.		Navigate To	F3					
	33 ENDCLASS.		Navigate to Declaration	Shift+F3					~
	<		Open in Project	Ctrl+Alt+P >					>
< >	*Global Class Class-relevant Lo	cal 1	Open Others	Alter Childer M/ S					
E Outline X	🖹 Problems 🗙 🔲 Properties 🛅	Ten	Quick Outline	Ctrl+0	er			78	- 0
CALC EPEE SEATS FOR FLIGHT	0 items	^	Cut	Chilly Y			-		
CRECTREESERIST ORTEROT	Description		Conv	Ctrl+C		Location	Туре		
			Paste	Ctrl+V	-				
		54	Get Where-Used List	Ctrl+Shift+G					
		_	Rename	Alt+Shift+R	L				
			Change Package Assignment						
			Quick Fix	Ctrl+1					
			Source Code	Alt+Shift+S >	Toggle Co	omment		Ctrl+7	
		淅	Activate	Ctrl+F3	Add Com	ment		Ctrl+<	
		23	Toggle Active and Inactive Version		Remove (Comment		Ctrl+>	
		2	Unlock	Ctrl+U	import Al	SAP DOC from De	scriptions		
		11.	Share Link for Selection		Format			Shift+F1	
		0	Coverage As	>	Delete un	used variables (a	D	Alt+U	
			Coverage As	>	Delete un	used variables (s	election)	Alt+Shift+U	
		0	Run As	>	Extract M	ethod		Alt+Shift+M	
		*	Debug As	>	Rename			Alt+Shift+R	
			Team	>	Occurren	ces in File		Ctrl+Shift+U	
Project: BTP-Trial-2022-10	v	Vrita	Compare With	Ctrl+Alt+C >	Show Co	de Element Inforr	nation	F2	10
			D I MOI	curració y	Show Me	thod Information		Alt+F2	

Figure 12 Renaming Method Parameters

First, the code must be saved. If this has not been done, a save prompt will appear, which must be confirmed.



Figure 13 Query Dialog for saving the Code



A dialog box appears for entering the parameter name.

듣 Rename Field			-	- 0	×
Choose Name					
New Name: * carrier_id				20 ren	naining
?	< Back	Next >	Finish	Car	ncel

Figure 14 Entering a new Parameter Name

If the object is already assigned to a transport, the renaming can be carried out directly with "Finish".

With "Next" further optional settings, such as the transport to be used and the activation option, can be made.

듣 Rename Fiel	d					×
Select a Trans	port Request				00	
Transport: Error Handling: Activation:	TRLK	d objects			Browse	
?		< Back	Next >	Finish	Cance	I

Figure 15 Transport Selection and Options



Before the final execution, a preview of the change can be displayed.

듣 Rename Field	— D X
Rename Field The following changes are necessary to perform the refactoring	ig.
Changes to be performed	
Z_FLIGHT_INFO==========CS (Global	Class)
🖅 BTP-Trial-2022-10	24 🐼 🕸 🖓
Original Source 8 IMPORTING i_f_carrier_id TYPE / 9 i_f_connection_id TYPE / 10 i_f_flight_date TYPE / 11 RETURNING VALUE(r_f_free_seats) TYPE i 12 PROTECTED SECTION. 13 13 PRIVATE SECTION. 14 15 >	Refactored Source 8 IMPORTING carrier_id TYPE / 9 i_f_connection_id T 10 i_f_flight_date T 11 RETURNING VALUE(r_f_free_seats) T 12 PROTECTED SECTION. I 13 PRIVATE SECTION. I 14 ENDCLASS. V V
?	< Back Next > Finish Cancel

Figure 16 Renaming Preview

After clicking on "Finish", both the parameters in the definition, the usage in the method and the parameter names are changed appropriately at the points where the method is used. The renaming shown here works analogously in the same way for variables, methods, and even class names.

This shows a very strong advantage over the SAP GUI-based development tools, in which a cross-object renaming is not possible automatically.

This is the first step, the basic functions and working methods in ADT are known and can now be applied. The first hurdle has been cleared and the basis for the application of the numerous functions of the ABAP Development Tools, which are explained in detail in the following sections, has been created.



3.2 Features of ADT

The main purpose of the previous section is to get you started and develop with the ABAP development tools in Eclipse. The Functions section is primarily dedicated to describing the numerous features and teaching best practices in dealing with ADT in daily work.

After ADT has been successfully set up and a project, i.e., a connection to a Netweaver On-Premise or an ABAP Cloud Environment in SAP BTP, has been established, development can begin. In the following chapters, we offer an overview of the possibilities that ADT offers in the development of new objects and in the extension/revision ("refactoring") of existing objects.

3.2.1 Overarching features

3.2.1.1 Workspaces

The so-called workspaces serve as the main level of work structuring and storage of the Eclipse and ADT configuration. When you start Eclipse for the first time, you will be asked in which directory the workspaces should be stored.

듲 Eclipse IDE Launcher	×
Select a directory as workspace	
Eclipse IDE uses the workspace directory to store its preferences and development artifacts.	
Workspace: F:\SAP-ABAP\ADT\WS-2023-03 V Browse	
Use this as the default and do not ask again	
Recent Workspaces	
Launch Cancel	

Figure 17 Query of the Workspace Directory

One way in which the directory structure can be designed can be found in Chapter 6 -Best Practices Eclipse Configuration.

Numerous configuration settings are stored in this directory. These include:



- the projects and thus accessible SAP systems,
- which favorite packages are used in the projects,
- which perspectives are used,
- which views and objects are open.

If you can get by with a workspace, you can check the box "Use this as default ..." so that this workspace can be used in the future without being asked. This setting can be changed at any time in the settings.

For most cases, one workspace is sufficient. If you work in several projects with different system lines or with different customers, the workspaces can help to keep the system environment used clear and to have the most efficient configuration available for every situation.

If there is a need to create or change a new workspace,

 $\mathsf{File} \to \mathsf{Switch} \ \mathsf{Workspace}$

either select a previously open workspace from the list or use

 $\mathsf{File} \to \mathsf{Switch} \ \mathsf{Workspace} \to \mathsf{Other}$

the workspace dialogue.



New Alt+Shift+N > Open File Open Projects from File System Recent Files Close Editor Close Editors Close All Editors Ctrl+Shift+W Save Save Ar.	- • • • // •
Open File Open Projects from File System Recent Files Close Editor Close All Editors Ctrl+Shift+W Save Ctrl+S	
Open Projects from File System Recent Files Close Editor Ctrl+W Close All Editors Ctrl+Shift+W Save Ctrl+S Save Ctrl+S	
Recent Files Close Editor Close All Editors Ctrl+S Save Ctrl+S	
Close Editor Ctrl+W Close All Editors Ctrl+Shift+W Save Ctrl+S Save Ar	
Close All Editors Ctrl+Shift+W Save Ctrl+S Save Ar	
Save Ctrl+S	
Save As	
Jave Asin	
Ctrl+Shift+S	
Revert	
Move	
🛃 Rename F2	
Refresh F5	
Convert Line Delimiters To >	
Print Ctrl+P	
🚵 Import	
Export	
Properties Alt+Enter	
Switch Workspace > F:\SAP-ABAP\ADT\WS-	2023-03
Restart F:\SAP-ABAP\ADT\WS-	Second
Exit Other	

Figure 18 Switching the Workspace



Working with ADT

듣 Eclipse IDE Launcher		×
Select a directory as workspace		
Eclipse IDE uses the workspace directory to store its preferences and develop	ment artifacts.	
Workspace: F:\SAP-ABAP\ADT\WS-2022-06	~	Browse
✓ Recent Workspaces		
<u>WS-2022-06</u>		
<u>WS-2023-03</u>		
<u>WS-Second</u>		
✓ Copy Settings		
Workbench L <u>a</u> yout		
Working <u>S</u> ets		
✓ Preferences		
?	Launch	Cancel

Figure 19 The Workspace Dialog

If you want to create a new workspace, enter a new name in the Workspace field. This will create a new workspace with the current settings after clicking on Launch. You define which settings are to be transferred from the source workspace using the Copy Settings. Alternatively, you can access one of the displayed workspaces under Recent Workspaces directly by clicking on the blue links.

3.2.1.2 Project Explorer

The Project Explorer is a central component for navigation in the integrated systems. As soon as you integrate a new system as an "ABAP Project", it appears in the list. If you log in to a system and expand it, you will receive further information about shared objects, favorites, inactive objects, etc., depending on the system. These repository trees can be freely defined and customized. At the package level, the view behaves like the SE80 and maps object hierarchies through which you can navigate.

3.2.1.3 Favorite Packages

For day-to-day work, it is advisable to add packages as favorites in which you work regularly, or which fall within your personal responsibility. This gives you a good overview and allows you to quickly find "your" objects.



Working with ADT

e ws_adt_dsag_cloud - e	clipse	IDE		—		>	×
File Edit Navigate Search	Pro	oject Run Window Help					
i 📩 🗝 📄 👘 i 🚍 i 🗙 i 💋	9	🗾 👫 🖏 🖨 🖗 🗉	n i 🕸 🕶 💽 🕶 🚱 🕶	🂁 🕶 🛷 🕶			
월 - 월 - 선 -가 수 -	<			Q	: 😭	\$	4
Project Explorer X	CDS	Navigator 🗖 🗖	1				
							_
v 🛃 CLOUD_ABAP-202210 [1	TRL, 1	00, EN]					
🗸 🖈 Favorite Packages (2	2.727)						
> 🖶 ZLOCAL (22.727)		New	>				
Eavorite Objects (0)	ß	Duplicate Tree					
> 🌣 Released Objects (3.1		Open ABAP Development Obj	ect				
	Þ	Сору	Ctrl+C				
	ж	Delete	Delete				
	₽	Add Package					
<	Ş	Refresh	F5				
🗄 Outline X	Q.	Coverage As	>	💵 B 🔝 F 具	T		
There is no active editor that p		Coverage As	>			P	000
	0	Run As	>	~			_
	蓉	Debug As	>				_
		Profile As	>				_
	>	Expand Tree by					
		Configure Tree	Alt+Enter				
	_			-			
			<				>
ᄸ [TRL] Favorite Packages							0

Figure 20 Adding Packages to Favorites

For storing and organizing favorites at object level, the "ABAP Favorites" plug-in can be recommended, which can be installed in Eclipse via the plug-in installation mechanism, see Chapter 7 - Plug-ins.





Figure 21 Project Explorer Detail Screen with Button Bar

In the button bar there are still some useful functions with which you can make settings on the view:

- Double arrow (Link with Editor) Objects that are in focus in the editor are displayed in the Project Explorer by loading the hierarchy.
- Three dots (View Menu) Further settings of the views, for example to create working sets. These are folders that can be used to group systems (see screenshot above).

3.2.1.4 Working Sets

If you are a developer working with multiple system lines, we recommend using the Working Sets. These make it possible to group projects in Eclipse and thus clearly display several systems.

The three-dot icon in the upper right corner of the Project Explorer offers various setting options. Among other things, the working sets can be created and configured here.



🎦 Project Explorer 🗙 🍕 Relation Explorer				
 	\$ 7	000	Projects Presentation Top Level Elements	> >
> 🔅 Released Objects			Select Working Set	
			Deselect Working Set Edit Act	
		Ô	1 Window Working Set	
		2	2 DSAG-ADT-Guide	
			Package Presentation	>
		\checkmark	Show 'Referenced Libraries' Node	
		7	Filters and Customization	
		\$ <u>1</u>	Link with Editor	

Figure 22 Working Sets Settings

The dialogue can now be used to create working sets (New) and make assignments (Edit).



Figure 23 Creation and Processing of the Working Sets

The edit button can be used to assign the desired projects to the working set in the subsequent dialogue.



Working with ADT

듣 Edit Working Set	—		×
Resource Working Set Enter a working set name and select the working set resources.			
Working set name:			
TEST-WS			
Working set contents:			
Select All Deselect All			
? Fin	ish	Cano	el

Figure 24 Assignment of Project to Working Set

This function enables a clear structuring of the systems according to system landscape or, if applicable, according to project or customer. Finally, the display of the Top-Level Elements must be set to Working Sets.



Figure 25 Project Explorer Display Setting





Figure 26 Illustration of Projects in Working Sets

3.2.2 Search and Navigation

The search for objects in Eclipse is a central part of the daily work, as is the navigation between the objects or the forward navigation. In this section, you will learn more about searching and navigating between ABAP objects.

3.2.2.1 Searching for objects

To search for or open an object in the system, you can use the "Open ABAP Development Object" dialogue (accessible via the key combination **CTRL+SHIFT+A**).



Working with ADT

🖨 Open ABAP Development Object 🛛 🗆 🗙
Project: * Browse
Enter search string and/or filter criteria (see help for details) 8
×
<u>M</u> atching items:
CONVERT_TO_LOCAL_CURRENCY (Function Module) Tr
G CX_RAP_QUERY_PROVIDER (Global Class) RAP Query Pr
OVDM (Annotation Definition) CDS annotations of the Vi
SEMANTICS (Annotation Definition) CDS Annotations fc
SEMANTICS (Knowledge Transfer Document) Document
G CL_MESSAGE_HELPER (Global Class) Helper Class for Sto
G CX_SY_CONVERSION_UNKNOWN_UNIT (Global Class) C
G CX_SY_CONVERSION_ERROR (Global Class) System exce,
ABAP (Structure) ABAP/4 language-internal type declar
F NROBJ (Authorization Field) Number Range Object
[F] NROBJ (Restriction Field) Number Range Object
< >
CONVERT_TO_LOCAL_CURRENCY
OK Cancel

Figure 27 Object Search Dialog

In the dialogue, you have the option of changing the ABAP project in the upper part and thus, deciding on which system you want to search for the object. If nothing has been entered in the search field, you will receive a history of the most recently opened objects. Use the *question mark* at the bottom to get more information, tips, and tricks on how to use the search. Detailed information on the object search can be found in the User Guide.

3.2.2.2 Filter Objects

In the object search, you now have the option of working with search strings and patterns to further narrow down the result set. The field supports "Content Assist" (CTRL+SPACEBAR) to use further restrictions and filters. A simple search might look like this:



nr*	×
Matching items:	
F NROBJ (Authorization Field) Number Range Object	^
[F] NROBJ (Restriction Field) Number Range Object	
[T] NROBJ (Restriction Type) Number Range Object	
Workspace matches	
NR (Message Class) Number ranges	
O NROT1F01 (Include) Include NROT1F01	¥
< >>	
More than 50 results	

Figure 28 Results of the Search

However, more than 50 results are displayed (the default) and most likely the desired result is not included in the result set. In this case, you can go to the "Content Assist" to get more options for filtering.

nr*	
<u>M</u> atch	🗊 type (type)
🖲 N	G group (Object Type Group)
[F]N	🖶 package (package)
[T]N	appl (Application Component)
Ô	🝦 owner (owner)

Figure 29 Display of additional Search Options

For example, if you want to restrict to table types, then you would restrict further by type (TYPE). The "Content Assist" also suggests the different types of objects, so that you can also find the table type (TTYP).

nr* type:ttyp	×
<u>M</u> atching items:	
P NR3SELT (Table Type) Auxiliary Type for Translation	^
P NREL_T_OT (Table Type) Migration Wizard: Outtab for I	
P NRIV_CALLS_TT (Table Type) Callstack for NUMBER_GE	
P NRIV_DOCU_TT (Table Type) Non-Assigned Numbers i	
P NRIV_TT (Table Type) Number range intervals	
P NROBJECT_RANGE_TT (Table Type) Range Table for Ot	~
< >	
i 7 results	

Figure 30 Result with Object and Type Filter



The other filters and types can be freely combined with each other to find the desired target quantity or object. By double-clicking on the desired entry, the object is then displayed in the editor.

To speed up the search, it is recommended to specify the type of the desired object, otherwise the search – compared to the usual speed in the SE80/SE11/etc. – takes a very long time.

3.2.2.3 Navigation

In the ABAP Workbench, you can navigate to the next object by double-clicking the corresponding expression in the source code. In Eclipse, only the source code is selected here. There are three ways to trigger forward navigation:

- Place the cursor on the object, press F3
- Hold down the **CTRL** key and click on the object
- A clickable link is offered in the interface (e.g. data element \rightarrow domain)

The object opens in a new tab within the editor, the source object remains open, and you can easily navigate between the most recently modified objects using the keyboard shortcuts:

- ALT+Right Arrow: forward
- ALT+Left Arrow: backward

This can also be done analogously with the arrow keys in the area of the pushbutton bar. In the area of the pushbutton bar, there are also various options to get to the last tab used (ALT+arrow left).

Figure 31 Navigations Icons



3.2.2.4 Displaying the ABAP Repository Tree

After you have found an object, in many cases you will want to continue working or researching in this package. To do this, you can load the object tree by activating the double arrow ("Link with Editor") in the Project Explorer.

Figure 32 Actions for the Project Explorer

In this case, the package hierarchy is loaded to the object focused on in the editor. You can then navigate to the other objects and structures in the Project Explorer.

The work with the repository tree is described in detail in the user guide.

3.2.3 ABAP Editor

The ABAP Editor is a simple text editor that enables the purely textual creation of ABAP artifacts. The context function can be used to call up the most important functions such as quick fixes, refactoring functions, and formatting functions. Getting started with the ABAP Editor is described in more detail in the section Creating a Class in Text Mode.



Working with ADT

```
C [TRL] Z_FLIGHT_INFO X
```

```
▶ G Z_FLIGHT_INFO ▶
   1⊖ CLASS z flight info DEFINITION
  2
       PUBLIC
  3
       FINAL
       CREATE PUBLIC .
  4
  5
       PUBLIC SECTION.
  6
         "! <H1>Feature to calculate free sets for a single flight</H1>
  7
         "! And here could be added more information about the class
  8
         • 1
  9
         "! @parameter carrier_id
                                    ID of the <strong>airline</strong>
  10
         "! @parameter connection id | ID of the <em>flight number</em>
  11
         "! @parameter flight_date
                                   12
         "! @parameter free_seats
  13
         METHODS calc_free_seats_For_flight
  14
          IMPORTING carrier_id TYPE /dmo/Carrier_id
  15
                    connection id
  16
                                     TYPE /dmo/connection id
                    flight date TYPE /dmo/flight date
  17
  18
           RETURNING VALUE(free_seats) TYPE i.
  19
       PROTECTED SECTION.
  20
       PRIVATE SECTION.
 21
     ENDCLASS.
  22
Global Class Class-relevant Local Types Local Types Test Classes Macros
🖹 Problems 🗙
                🔲 Properties) 🛅 Templates) 💷 Bookmarks) 🔊 Feed Reader) 블 Transport Organizer
0 items
```

Figure 33 ABAP Editor - Main Window

3.2.3.1 Element Info

By positioning the cursor on an object and the shortcut **F2**, a pop-up with additional information appears. Here is an example of a method and a data element:

189	METHOD if oo adt classrun~ma	in.		
19 20 21	out-swrite(Text aut der ENDMETHOD. ENDCLASS.	main importing out type ref to if_oo_adt_classrun_out	^	
			~	
<u>G</u> lobal	< <u>Class</u> <u>Class-relevant Local Types</u> <u>Local Types</u> <u>T</u> est Class		•	

Figure 34 Element Info for a Method



8	PARAMETERS: p_this TYPE keep	ennzx.	
9 10		kennzx type c length 1	~
		Documentation	
		Kennzeichen	
		\$	
		· ↔ ↔ ↔ ☆ ☆ ⊯ Ē	:

Figure 35 Element Info for a Data Element

In the Element Info, details are thus directly visible for which you still had to navigate the system in the old world. In addition, you can navigate further in the Element Info to view details of deeper levels, e.g. the stored domain, etc.

The ABAP Element Info also exists as a stand-alone view. This can be activated via the Windows \rightarrow Show View menu \rightarrow ABAP Element Info. After that, "Link with Selection" automatically displays the Info element and the documentation from ABAP Doc each time a relevant development object is clicked:





Figure 36 Display of the Element Info after Selecting the Object

Via "Pin this view" the \square information is displayed permanently, even if another element is clicked or the element info is opened by **pressing F2** for another development object.

3.2.3.2 Source Code Formatting with the ABAP Formatter

In the SAP GUI, the tool for formatting the source code is called Pretty Printer. The counterpart in ADT is the ABAP Formatter. It can be accessed either via the keyboard shortcut

SHIFT+F1

or via the context menu in the source code.



	B	Paste	Ctri+v		
1PL	e.	Get Where-used List	Ctrl+Shift+G		
0.00		Rename	Alt+Shift+R		
	₩,	Change Package Assignment			
		Quick Fix	Ctrl+1		
as		Source Code	Alt+Shift+S >	Toggle Comment	Ctrl+7
	*	Activate	Ctrl+F3	Add Comment	Ctrl+<
	22	Toggle Active and Inactive Version		Remove Comment	Ctrl+>
	2	Unlock	Ctrl+U	Import ABAP Doc from Descriptions	
•	11	Share Link for Selection		Format	Shift+F1
	9	Coverage As	>	Format Block	Ctrl+Shift+F1

Figure 37 Formatting Context Menu

For the ABAP Formatter to be able to do its job, it must be configured in advance. Analogous to the Pretty Printer settings in the SAP GUI, you specify whether indentations are to be made and how the upper and lower case is to be formatted. This is done separately for each ABAP project.

3.2.3.2.1 Setting up the ABAP Formatter

If the settings have not yet been made, a pop-up window will appear with the message that this must be done first. There is also a link to the settings included in this pop-up window. Alternatively, you can also use them directly via the context menu entry *Properties* of the corresponding project. In the image below, you can see the location in the settings.

The settings correspond to those in the SAP GUI. If you try out the individual options, you will see the respective result in the preview window. A new feature is the ability to retain camel case identifiers. This is particularly practical in connection with the CDS views, as they are consistently used in the virtual data model of SAP (VDM).



Properties for EHX_100_d0000472203_en		- D X
type filter text	ABAP Formatter	
	Indentation Indent Lines Format Functional Method Calls Condense One-Line-Method Call Upper/Lower Case Conversion None Derived from First Statement Custom Keywords Upper Case Lower Case Comel Care Identifier	↓ Identifiers ○ Upper Case ④ Lower Case
Run/Debug Settings Semantic Resource Service Policies Task Tags > Validation WikiText	Preview LOOP AT Cars INTO DATA(Car->drive(speed_value = 130 sp ENDLOOP.	Car). eed_unit = 'kph').
?		Apply and Close Cancel

Figure 38 Upper and Lower Case in ABAP Formatter

3.2.3.3 Quick Fixes

Quick fixes are automated solutions for common problems in the context of software development with ABAP. Quick fixes are offered natively by ADT, but can also be extended by various plug-ins. The use due to the large number of available Quick Fixes makes working with ADT much more efficient than with the SE80. In addition, the risk of errors due to manual adjustments is reduced. ADT make the adjustments automatically and always identically.

Quick fixes provide functionality for two areas:

- Automatic creation of non-existent objects (e.g. method implementations)
- Automatic modification of existing objects without changing the functionality (socalled refactoring, e.g. "extract method")



11	DATA: clas	<pre>s_runner TYPE REF TO if_oo_adt_classrun.</pre>	
12	ENDCLASS.	⊯ Rename class_runner (Ctrl+2, R)	Renames class_runner and adjusts all occurrences of class_runner in the current source unit
13		Generate Getter for class_runner	If class_runner is also used in other source units, the rename wizard will be
14		Generate Setter for class_runner	opened.
15		Generate Getter and Setter for class_runner	
169	CLASS zcl main	Make class_runner protected	
179	METHOD const	Make class_runner public (read-only)	
1/0	METHOD CONST	Make class_runner public	
18	class_runr	X Delete class runner	
19	ENDMETHOD.	-	
20			
21	ENDCLASS.		
		Press 'Ctrl+Shift+1' to show in Quick Assist View	Press 'Tab' from proposal table or click for focus

Figure 39 Display of Refactoring Options

Due to the large number of quick fixes and the constant changes in this area, the individual quick fixes are not described here. An overview can be found in the documentation.

3.2.4 Other object types

3.2.4.1 Programs and function groups

Programs and function groups are displayed in the *Source Code Library* folder in the navigation of the Project Explorer. Under the corresponding object types, all components are displayed according to the SAP GUI.



Figure 40 Comparison of Function Groups in the Project Explorer of ADT and in the SE80



The same source code editor is used as for ABAP classes. This means that all features from ABAP Formatter to Language Help are mostly identical.

3.2.5 ABAP Views

3.2.5.1 Outline

The Outline View provides information about the currently focused development object and resolves variables, local classes, types, etc. The view can be compared with the SE80, but only shows you the current context of the object. In the screenshot you can see a class, below with the corresponding methods and two private attributes of the class. By clicking on an entry, you navigate to the corresponding location in the source code.



Figure 41 Display of Properties in the Outlines

In the button bar there are other different functions for the view:

- "Sort" sorting of entries by alphabet or by type and alphabet
- "Hide Non-Public Members" Only attributes and methods that can be used from the outside are displayed (topic visibility)

Note: In Chapter 7 - Plug-ins you will find more information about the plug-in "Classic Outlines", which extends the outlines.



3.2.5.2 Problems

The Problems View is probably one of the most important views. It shows information about errors within development objects in the form of a list. The view updates automatically when new errors are added, or existing ones are corrected. A continuous "checking" of the source code is not necessary.

In the standard configuration, all errors in the user's own open objects, not just the object currently in progress, are displayed across all systems.

👔 Problems 🗙 🔲 Properties 🛷 Search 🚡 Templates 💷 Bookmarks 🔊 Feed Reader 📮 Transport Organizer 🌄 AB	AP Unit 💡 Quick Assist 📮 Console 🙁	Variables 指 ABAP Ty	pe Hierarchy 📃	ABAP Element Inf
2 errors, 0 warnings, 1 other				
Description	Resource	Path	Location	Туре
✓ S Errors (2 items)				
📀 "X" konnte nicht interpretiert werden. Mögliche Fehlerursachen: Falsche Schreibweise oder Kommafehler.	zprg_adt_test_report.asprog	/ISH_100_dpanzer	line 8	ABAP Syntax
😭 Type "ZCL_CLASS_RUN_DEMO2" is unknown.	zcl_main_logic.aclass	/TRL_EN/.adt/classl	line 18	ABAP Syntax
✓ i Infos (1 item)				
ᢪ Die Paketprüfung ist ausgeschaltet	zprg_adt_test_report.approg	/ISH_100_dpanzer	Unknown	ABAP Packag

Figure 42 Displaying the Messages in the Problems View

By double-clicking you can navigate to the corresponding place in the source code.

The button ⁸ can be used to further configure the view:

	Show	>
	Group By	>
	Sort By	>
	New Problems View	
7	Filters	
	Configure Columns	

Figure 43 Display of the Options of the View

- Under "Show" you can configure which errors/warnings are displayed in the Problem View, e.g. only those of the currently in progress development object or all of them.
- "Group by" allows grouping according to various criteria, usually grouped by "Severity", i.e. error/warning/info.
- "Sort By" can be used to change the order of the display.
- "New Problems View" duplicates the view.



- "Filters" allows you to filter the list of results in detail down to the development object type.
- "Configure Columns" allows you to show and hide columns and change the order of the columns.

Filters	_	_		×
Configurations				
Errors/Warnings on Project New Errors/Warnings on Selection Remove All Errors on Workspace Rename On selected elements in selected projects On selected elements only On selected elements only On selected elements and its childre On working set: Window Working Select Vertex Vertex Contains Vertex All Errors Warnings Select Vertex Contains Vertex Contains Vertex Andp Syntax Check And Buildfile Problems Andp Syntax Check ODIC Check Problem DDIC Check Problem VUse limits Vertex	en Set	nfo Sel	lect All elect All	
Restore Defaults Apply an	d Close		Cance	

Figure 44 Configuration of the Displayed Points

3.2.5.3 Properties

The Properties View collects information that is displayed in the classic SAP GUI under "Properties". These include, for example:

- State development object (active/inactive)
- Creation and modification information



• Package Mapping

\square Properties $ imes$	🛷 Search	눰 Templates	🛄 Bookmarks	🔝 Feed Reader	🖵 Transport Organizer	🗸 ABAP Unit
Selection from [TR	L1 ZCL MAIN	I LOGIC				

General Package: ZDSP -> ZLOCAL Specific API State Version: Active Transport Description: Hauptlogik CB9980000948 (11b1e759-45a9-4493-ba86-5bab44666f55) Responsible: Created on: Monday, September 19, 2022 Last changed by: CB9980000948 (11b1e759-45a9-4493-ba86-5bab44666f55) Last changed on: Monday, September 19, 2022, 12:37:52 PM Original language: EN Original system: TRL Application component: ABAP language version: ABAP for Cloud Development

Figure 45 Properties View

In the "Specific" area, object-specific properties are displayed and can be partially changed there. For classes, for example, the fixed-point arithmetic can be set here.

In the Transport section, a history of the transport orders in which the object is contained is listed.

O [TRL] 7	G [TRL] ZCL_MAIN_LOGIC										
Connect	Unreleased Transport Requests (1)										
Specific	Request	Description	Owner	Created	Changed	# Objects	# Entries	Status			
API State	TRLK901142	Dummy	CB9980000948 (11b1e759-45a9-449	1 day ago	1 day ago	3	3	Modifiable			
Transport											

Figure 46 History of Transports

The context menu can be used to branch to the transport request for further analysis.



Request: TRLK901142 (Workbench)							🥝 具 賃 🕶 <i>11</i> (?
▼ Properties							
Short Description: * Dummy			Owner:	CB9980000948 (11b1e759-45a9	0-4493-ba86-5bab44666	f55)	Browse
Long Description:			Target:		Browse		
			CTS Project:				Browse
			Status: Last Changed: Source Client:	Modifiable Monday, September 19, 2022, 100	10:07:35 AM		
▼ Objects							
✓ ■ TRLK901142 - 11b1e759-45a9-4493-ba86-5bab44666f55	type filter text						
All Objects	Object Name OZCL_CLASS_RUN_DEMO OZCL_MAIN_LOGIC ZDSP	Descripti Class Run Hauptlog DSP	on Demo k	Object Type CLAS CLAS DEVC	Program ID R3TR R3TR R3TR	Lock/Import Status Dbject Locked Dbject Locked Dbject Locked	IMG Activity

Figure 47 Transport View

Depending on the type of open development object, there are additional areas that contain specific information about the respective object.

In order to view several objects in parallel, several properties views can also be created via the button.

The Properties View has several advantages compared to classic SAP GUI development:

- It is (usually) constantly displayed and available, so that there is no need to navigate time-consumingly.
- It aggregates information that was previously only identifiable via multiple transactions/tabs/tabs.

3.2.5.4 Templates

Code templates are ready-made samples of source code that can be implemented in any way in an application. These patterns reflect static source code and have dynamic elements in the form of variables. In the standard delivery of ADT, some templates are delivered.

TEMPLATE VIEW

Templates are made available via a separate view (Window \rightarrow Show View \rightarrow Templates) and can also be adjusted via the settings (General \rightarrow ABAP Development \rightarrow Editors \rightarrow Source Code Editors \rightarrow ABAP Templates).



Working with ADT

	ති	0	×	会 🗅	00
Name	Description		•• 1	~ ~	^
V 💷 ABAP					
assertEquals	assert_equals (ABAP Unit)				
case	Case block				
📄 catchAll	##CATCH_ALL pragma (unhandled exception of				
🗐 functionModuleParame	Function Module Parameter Block				
📄 if	If block				
📄 ifElse	lf-else block				
E Icl	Local class				
📃 lif	Local interface				
IoopAtAssigning	Loop-at-assigning block				
needed	##NEEDED pragma (superfluous statements)				
📄 noHandler	##NO_HANDLER pragma (unhandled exception				~
Q Preview					
<pre>if \${expr}. \${word_selection}\${curs else. \${elseblock} endif.</pre>	or}				<

Figure 48 Template View Browser

The view consists of a button bar in the upper area, a list of code templates and a preview of the template in the lower part.

USE TEMPLATE

The template can be used directly in the source code. Start typing the name and select the appropriate template with the help of the "Content Assist" (here are the first two entries).





Figure 49 Selection of Templates in Content Assist

The entire template is inserted, and you can start filling in the placeholders (variables). Use the tab to jump back and forth between the individual placeholders.

Frequently used templates for use are, for example:

- Icl plant of a local class
- testClass Creation of a test class
- **functionModuleParameter** Example interface for function modules

The creation of your own templates is very well suited for saving development effort for repetitive tasks or similar code sections. Furthermore, they can be helpful in training if you want to insert larger sections of code without copy-and-paste.

CREATE TEMPLATE

In principle, a template can be defined with all language commands used in ABAP (source code, comments). Variable parts of the template can be tagged with placeholders ("\${placeholder}"). Variables are also available for deriving context-specific information. These are available for the following scenarios:

- Name of the object
- Name of the package
- ID des Systems
- User, date and time
- Current year
- Cursor position after insertion



Placeholders with the same name are always uniformly adjusted after insertion (e.g. the name of the class).

AVAILABILITY OF TEMPLATES

Templates are available within an Eclipse workspace, but unlike your own patterns, they are available across systems. Templates can be imported and exported via the settings in order to exchange them among colleagues/employees. It is not possible to store templates centrally for all developers.

3.2.5.5 Where-Used-List

The where-used list finds all static uses of a development object in the source code of the current project. The where-used list can be found using the key combination

```
STRG+SHIFT+G (Get-Where-Used-List)
```

attainable. The result is displayed in the "Search" tab:



Figure 50 Result of the Where-used List

By using the filter function with the help of the filter icon, it is possible to restrict to packages, object types and users. Here, too, auto-completion can be used via **CTRL+SPACE** to find objects faster.



Filter		_		×
Filter Where-Used				
(i) Enter Filter Criteria				
	0.			
Package(s):	×			
Object Type(s):				
Responsible User(s):				
Code Category:	Production and Test Code			\sim
Exclude Indirect	References			
		_		
(?)	Apply Reset		Cance	ł

Figure 51 Filter for Where-Used-Search

Via **CTRL+.** and **CTRL+**, the citations can be conveniently browsed, i.e. the next or previous citation can be displayed.

3.2.5.6 Bookmarks

Bookmarks are "links" to development objects defined in the system. There are often key points where adjustments are often necessary. There can be various reasons for this, for example:

- A large historically grown include, where extensions take place again and again.
- A class with complex logic that turns out to be error-prone.
- Objects that are regularly worked on.

Bookmarks can be created by right-clicking on the list next to the source code:



	1	6⊖ (CLASS zcl_main_log	gic IMPLEMENTA	TION
	1	7⊝	METHOD construct	tor.	
	1	•	-1	NEU1 -1	n nu
	1	۲	Toggle Breakpoint	Ctrl+Shift+B	_
	- 1	0	Toggle Soft Breakpoint	Alt+B	
	2		Disable Breakpoint	Shift+Double Click	
	4	Ş	Refresh Breakpoint Activation	Ctrl+Alt+B	
۷	* <u>G</u> I		Debug Properties		<u>M</u> acr
-			Breakpoint Properties	Ctrl+Double Click	narks
4	0 ite		Add Bookmark		
_	De		Add Task		
		✓	Validate		
		\checkmark	Show Quick Diff	Ctrl+Shift+Q	
		~	Show Line Numbers		-
			Folding	>	
			Preferences		

Figure 52 Menu for Creating the Bookmark

By default, the code of the selected line is specified as the name. It is a good idea to assign a descriptive and technically meaningful name, which can then be used to easily find the bookmark again.

~ `

Figure 53 Entering a Name (Bookmark)

Within the source code editor, a small blue flag now appears next to the selected line number:



16⊖CLASS zcl_main_logic IMPLEMENTATION.									
17⊝	METHOD constructor.								
18	<pre>class_runner = NEW zcl_class_run_demo().</pre>								
19	ENDMETHOD.								
20									
21	ENDCLASS.								

Figure 54 Display of a Bookmark in the Source Code

The bookmark is then available in the list and can be accessed by double-clicking. You can also customize the view and perform filtering via ⁸. The context menu allows you to delete the bookmark and edit the description.

💷 Bookmarks 🗙 🔝 Problems 🔲 Properties 🔗 Search 📔 Te	emplates 🔊 Feed Reader	具 Transport	Organizer	🔽 AE
1 items				
Description	Resource	Path	Location	
Initialisierung der Prüfung der Stammdaten des Geschäftspartners	zcl_main_logic.aclass	/TRL_EN/	line 18	

Figure 55 Bookmarks View

3.2.5.7 Sharing ADT-Links

In everyday developer life, it often happens that code has to be talked about together (e.g. during reviews) or a problem is found in a piece of coding that is the responsibility of another developer (no shared code ownership). Often it is then said "Can you please take a look at the class *XYZ method ABC line 1203* ... I think there's a bug?". The other developer has to navigate through the IDE until he finds the mentioned place.

ADT offers the possibility to send a link that leads the recipient directly to the appropriate code location when he clicks on it. To do this, an area must be selected in the source code and then selected "Share Link" in the context menu.


\triangleleft	Undo	Ctrl+Z
	Revert File	
	Save	Ctrl+S
î:	Open ABAP Type Hierarchy	F4
	Quick Type Hierarchy	Ctrl+T
	Navigate To	F3
	Navigate to Declaration	Shift+F3
	Open in Project	Ctrl+Alt+P >
	Open With	>
	Open Others	>
	Show In	Alt+Shift+W >
	Quick Outline	Ctrl+O
of	Cut	Ctrl+X
Ð	Сору	Ctrl+C
Ē	Paste	Ctrl+V
Æ	Get Where-used List	Ctrl+Shift+G
	Rename	Alt+Shift+R
#	Change Package Assignment	
~	Quick Fix	Ctrl+1
	Source Code	Alt+Shift+S >
₩	Activate	Ctrl+F3
23	Toggle Active and Inactive Version	
8	Unlock	Ctrl+U
11	Share Link for Selection	
0	Coverage As	>
	Coverage As	>
0	Run As	>
*	Debug As	>
8	Profile As	>
-	Team	>
	Compare With	Ctrl+Alt+C >
	Replace With	>
	Validate	
	Preferences	

Figure 56 Sharing the Source Code as a Link (Context Menu)



🖨 Sha	e Link			×
Select a	link type from the list			
	HTTP Link To open a development object in a web browser. Good for sharing v	/ia email		
Å	ADT Link To open a development object in ABAP Development Tools			
?	Email link (Copy link	c to clipb	oard

Figure 57 Link Sharing Dialog

The correspondingly generated link can then be sent to the colleague by e-mail or copied to the clipboard and sent via a chat program, for example. You have the choice between HTTP link and ADT link. HTTP links are opened directly in the browser, ADT links branch to ADT (Eclipse).

Structure of an ADT link (URI):

adt://<System>/sap/bc/adt/oo/classes/<Klasse>/source/<Methode>#start=18,0

More details can be found in the user guide .

3.2.5.8 ABAP Type Hierarchy

The View Type Hierarchy is used to represent the inheritance hierarchy of classes and interfaces. To use the view, all you have to do is place the cursor on the class or interface and press the shortcut **F4**. Alternatively, you can open the ABAP Type Hierarchy via the context menu.



0 /	/DMO/IF_FLIGHT_LEGACY >			
1	"! Interface for Fl:	ight	Legacy Coding 	
2	"! Every used structure or	table	e type needed in the API Function M	odules
3	"! will be defined here.			
4⊝	INTERFACE /dmo/if flight_le			
5	· PUBLIC.	$\langle \mathcal{Q} \rangle$	Undo	Ctrl+Z
6			Revert File	
7	********		iter citerine	
8	* Version information *		Save	Ctrl+S
9	*******			
10	CONSTANTS co_version_majo	ង	Open ABAP Type Hierarchy	F4
11	CONSTANTS co_version_mino		Quick Type Hierarchy	Ctrl+T
12			Navigate To	F3

Figure 58 Opening the ABAP Type Hierarchy

The view displays the hierarchy in a tree structure.

🍃 ABAP Type Hierarchy 🛛		월 원	₽ ₽	000	
[]/DMO/IF_FLIGHT_LEGAC	CΥ				
V DMO/IF_FLIGHT_LEG	ACY				
Ø /DMO/CL_FLIGHT_I	LEGACY				

Figure 59 Display of the Type Hierarchy in the View

You can double-click to open the selected object in the ABAP Editor. The Quick Type **Hierarchy** can also be opened inline in the code via **CTRL+T**.

3.2.5.9 Transport Organizer

The Transport Organizer displays an overview of the open ABAP projects. Under the systems, you will find the current transports in the system. By default, you can see all your own transports. By right-clicking on a system and selecting "Configure Tree ..." these settings can be overridden and transports can also be viewed by other developers.



📮 Transport Organizer 🗙			÷ 🖻 🖉
type filter text			
Transport Request	Owner	Туре	Description
✓ ☑ refreshed: 21:19:39, 02.11.2022]			
V 🗁 Workbench			
🗸 🗁 Local Change Requests			No Target # Release
🗸 🗁 Modifiable			
✓ ■ K900073	- REPRESENTATION - B-		Local RAP
🗸 📋 К900074	- REPRESENTATION D	Development/Correction	Local RAP
H YBS_ HEAR AND A		Package	Test für RAP Modell
H YBS_		Package	Einfache Demo
YBS_		Table	Simple data
YBS_		Technical Attributes of a Table	
D YBS_ ANALYSIS		Data Definition Language Source	Einfache View
D YBS		Data Definition Language Source	Simple View C
B YBS_		Behavior Definition	Einfache View
⊙ ZBP_		Class (ABAP Objects)	Behavior Implemen

Figure 60 Transport Organizer View

All functions of the Transport Organizer (SE09/SE10) are available from the SAP GUI:

- Double-click Show details of the job/task in your own view
- Right-click Various functions such as: change user, consistency check, share

3.2.5.10 Feed Reader

The Feed Reader can be used in conjunction with ADT to receive specific notifications from the SAP system. By default, the following notifications are consumed for an ABAP project:

- Runtime error (dump) caused by the user's own user
- Runtime error for objects for which the user is responsible
- Systemmeldungen

S Feed Reader ∞						🖉 🗟 🕶 👜 🗛 🏄 🖇 🗖 🗖	3
type filter text for feed entry						Tuesday, October 04, 2022 04:01 PM	
Project / Feed Query / Feed Entry	Date	Time	Author	Total	Unread	The current application has intentionally triggered a termination	
> 億				0	0	Bustime Errors equired by ma (
				0	0	Runtime Errors daused by me (
> 🗇 🗇				0	0	Show in Runtime Error Viewer	
> (5) (1) (1)							
	04.10.2022	16:02:30		2	0	Contents	
Runtime Errors caused by me (***********************************	04.10.2022	16:02:30		1	0	Header Information	
The current application has intentionally triggered a terminat	04.10.2022	16:01:23	1900 100	<i>K</i> .		What happened?	
> 4 Runtime Errors for objects I am responsible for (2000)	04.10.2022	16:02:30		1	0	Error analysis	
System Messages	04.10.2022	16:02:30		0	0	Source Code Extract	
> ()				0	0	Active Calls/Events	
> (3)				0	0	Header Information	
Native Feeds				0	0		
						Short Text I he current application has intentionally triggered a termination with a short dump.	
						Runtime MESSAGE_TYPE_X_TEXT Error	~

Figure 61 Representation of a Run-Time Error



S Feed Reader ⊠				🔗 📓 🕶 🚔 🗛 🕺 👘 🗖				
type filter text for feed entry						Tuesday, October 04, 2022 04:07 PM		
Project / Feed Query / Feed Entry	Date	Time	Author	Total	Unread	This is a test		
> 傳				0	0	- Official messages [
						This is a test		
> (5)				0	0			
> I N SVAR				0	0	Created on (System Timestamp)	04.10.2022 16:07:27	
Z Z	04.10.2022	16:07:31		3	0	Message ID	100	
> 🍇 Runtime Errors caused by me (👘 💭)	04.10.2022	16:07:31		1	0	Created by	201201	
> 🍇 Runtime Errors for objects I am responsible for 🖉 👘 👘	04.10.2022	16:07:30		1	0			
🗸 📮 System Messages	04.10.2022	16:07:30		1	0			
This is a test	04.10.2022	16:07:27	1000					
> 傳 () () () () () () () () () (
> 傳				0	0			
Native Feeds				0	0			

Figure 62 Example of a System Message

By clicking on the button "Add feed..." you can consume more notifications. The type of notification that is available depends on the release status of the linked SAP system. For most notifications, you can specify different filter criteria. In addition, you can also add a normal Atom or RSS feed via URL and thus consume it.

Details can be found in the user guide in the help functions of Eclipse. The article Getting Feeds is a good starting point.

3.2.5.11 ABAP Unit

After executing ABAP Unit tests, the ABAP Unit view opens and lists the executed test methods and their statuses. The ABAP unit tests can be started using the shortcut **CTRL+SHIFT+F10** or the context menu in the ABAP Editor with the menu item "Run As".

	Coverage As	>			
0	Run As	>		1 ABAP Application	F8
*	Debug As	> 0	₽	2 ABAP Application (Console)	F9
	Profile As	>	4	3 ABAP Test Cockpit	Ctrl+Shift+F2
	Team	>	~	4 ABAP Test Cockpit With	
	Compare With Ct	trl+Alt+C > 📘	U	5 ABAP Unit Test	Ctrl+Shift+F10
	Replace With	>	U	6 ABAP Unit Test With	Ctrl+Shift+F12
	Validate	Ð	2	7 ABAP Package Check	Ctrl+Alt+F5
	Preferences			Run Configurations	

Figure 63 Executing the ABAP Unit Test via the Context Menu

Within the view, you can filter the list by status, re-execute test cases and display details about faulty runs. The latter appears by clicking on the affected test method.



🚮 ABAP Unit 🛛					🖻 🕀 🍸 🔻 隆 🕶 🗒 🗖 🗖
[] ZCL_DMO_FLI	GHT_TEST (17:14:12)				
Methods: 1		Duration:	256 ms		
🗹 🗗 0	🗹 🏝 0	🗹 🖉 0	🗹 🔽	1	2 0
∽ 🚡 zcl_dmo_flig	🔳 Fai	ilure Trace			
✓	[harmless, short]				
test_cr	eate (< 0.01s)				

Figure 64 Display of the Results of the ABAP Unit Test

The scope of the test methods to be executed can be determined in various ways. On the one hand, it depends on the context. For example, if you focus on a single test method, only this method is executed. If the focus is on the class to be tested, then all test classes (and test methods) are executed. You can even extend the whole thing to a complete package by highlighting the package in the Project Explorer and running the unit tests. In addition, you can rerun individual or all tests in the view via the context menu - depending on which level you choose. For example, you could run all test methods of only one test class. This option is especially helpful if a test case is not successful and you need to analyze the behavior.



Figure 65 Restart the Execution

On the other hand, you can use "ABAP Unit Test With..." determine what kind of tests should be performed.



Preview or Execute Unit Tests Configure tests for ZCL_DMO_FLM	GHT_TEST
Scope ☑ Own Tests ☑ Foreign Tests	Additional Measurements Coverage Trace
Risk Level Harmless Dangerous Critical	Duration Short Medium Long
?	Execute Preview Cancel

Figure 66 Dialog for Setting the Execution of ABAP Unit Tests

Thus, for example, you could only execute the test methods in a package that have the risk level "Dangerous" and the duration "Medium".

3.2.5.12 ABAP Coverage

The ABAP Coverage view appears when you run ABAP unit tests with coverage. Test coverage provides an indication of which code is not covered by automated testing. The test coverage there can be a conscious decision, since a test coverage of one hundred percent requires a lot of effort in development in the long run. Coverage can also provide clues as to where more test coverage might be needed. A blanket recommendation for test coverage cannot be given and may also depend on the criticality of the application.

This type of execution can be started via shortcut **CTRL+Shift+F11** or via the context menu with the menu item "Coverage As".

	Coverage As >	U	1 ABAP Unit Test	Ctrl+Shift+F11		
0	Run As >		Coverage Configurations			
Figure 67 Carrying out the Coverage Measurement						



The view displays the executed code in a tree structure and provides information about how many statements were executed absolutely and relative to the executed test methods. In addition, the ABAP Editor color-coded exactly which statements were executed (with green) and which were not (with red).

22⊖ 23	CLASS zcl_dmo_flight_test IMPLEMENTATION.							
240	METHOD create travel.							
25	/dmo/cl flight legacy=>get instance()->cr	eate travel(
26	•••••• EXPORTING is_travel = VALUE #(agency_id	= '070001' customer_id =	'000001'					
27	begin_date = '20220101' end_date = '20220101' end_date = '20220105')							
28	<pre>IMPORTING es_travel = DATA(ls_travel)</pre>							
29	····).							
30	travel_id = ls_travel-travel_id.							
31	ENDMETHOD.							
330	METHOD remove travel							
34	/dmo/cl_flight_legacy=>get_instance()->de	lete travel(traval id).						
35	ENDMETHOD.							
36								
37	ENDCLASS.							
	<							
Global	Class Class-relevant Local Types Local Types Test Classes	Macros						
🚽 AB	AP Unit 🗎 ABAP Coverage 🔀		Z 8 D					
ZCL_DMO_FLIGHT_TEST (19.09.2022, 17:19:23)								
Eleme	lement Coverage Covered Statements							
~ O	✓ ⊙ ZCL_DMO_FLIGHT_TEST=======CP = 60.00%							
~	✓ ⓒ ZCL_DMO_FLIGHT_TEST ■ 60.00% 3							
	 CREATE_TRAVEL 100.00% 							
	REMOVE TRAVEL		0					
	-							

Figure 68 Color Highlighting of Source Code after Unit Test

If the colored view is not visible in the source code, it can be activated via the icon

3.2.5.13 ATC and Exemption

The ABAP Test Cockpit can be executed in ADT as well as in the SAP GUI. There are several ways in which you can start the exam:

- Using the keyboard shortcut **CTRL+SHIFT+F2**
- Right-click in the Project Explorer under the item "Run As"
- In the ribbon at the top, below the button to start the object

After the tests have been executed, you will receive the view for the "ATC Problems", i.e. the feedback on the messages found by the set tests.



		TC F	roblems	×	📑 🚉 8 🗗
▶ 🛃 → 🗟 ABAP_CLOUD_DEVELOPMENT_DEFA	ULT 🕨	ۯ Last Check	Run		
type filter text					Details
Description V 📴 Findings: 2 Infos	Lin	Contact Pe	Obje	Check	$\langle \neg \neg \rangle \stackrel{\text{de}}{=} A \stackrel{\text{d}}{A}$
Infos (2 Infos)					Appl. Comp. Check / Check Class / Message Code 💧
Strings without text elements are not translated	8 (IF_C	- 3099000000-1	CLAS	Extended Prog 2	BC-ABA-LA-EPC / CL_CL_TEST_EXTENDED_CHECK / 1700
Strings without text elements are not translated	14 (IF_	- (10)00000000000000000000000000000000000	CLAS	Extended Proc 2	
					Details of Analysis
					 Strings without text elements are not translated: 'Test Mail'
					Finding can be suppressed with pragma ##NO_TEXT
					What is checked?
					This check ensures that customers do not see any foreign language texts.
					Procedure
					It is possible to use the rules below to flag a character string as non-text-like (for example, by using only uppercase characters). As a final resort, it may sometimes be necessary to suppress the message using a pragma.
					string = 'Data Hugo TYPE I.' ##NO_TEXT .
٢				>	How is the check done?

Figure 69 Display of the Results of the ATC Run

On the left, the messages are displayed sorted by the severity of the error. On the right you will find information about the selected entry. Here it is explained again for you what has been checked and what a correction can look like. In the button bar at the top of the view, the result can also be deleted again. In this way, the markers in the source code will disappear.

By right-clicking on the message, an exception can also be requested via the menu item "Request Exemption".



Figure 70 Requesting Exemptions via the ATC View



The information of the form corresponds to the SAP GUI and guides you through the approval process. At the end, the request can be processed as usual via the ATC Cockpit.

Request Exemption			×
Exemption Scope			
Select the scope the exe	mption is valid for		
Finding			
Programm-Include:	(10000), 2017, 405, 50-		
Programm:	2000, No. 10, 10		
Package:			
Message:	Inkorrekter Pretty-Print-Zustand		
Check:	Überprüfung des Pretty-Print-Zustands		
Checksum:	1076177054		
Apply exemption to			
?	< <u>B</u> ack <u>N</u> ext > <u>F</u> inish	Cance	el

Figure 71 Dialog for Classifying the Exception

3.2.5.14 ABAP Language Help

In any source code editor, for example for ABAP, CDS, or BDL, the F1 key can be used to call the respective language help (not just ABAP!) directly for the keyword on which the cursor is located. Alternatively, you can also get it via the context menu by right-clicking on the corresponding instruction:

Source Code \rightarrow Show ABAP Language Help

This allows you to get support at any time if you are not sure of the exact syntax of a statement.



357 DATA(result) = R Reference Undo Wining Cui+Z Type int4 359 See Cui+S Ine IN tasks 360 See Cui+S Ine IN tasks 361 See Cui+S Ine IN tasks 362 Open ABAP Type Hierarchy Fd 363 Out ->write(data Open ABAP Type Hierarchy Fd 364 Out ->write(name See Cui+S Ine IN tasks 366 ENDMETHOD, Open Others See Cui+C 366 ENDMETHOD, Open Others See Cui+C 367 Open Others See Cui+C Open Others 368 Coty Cui+C Open Others 366 ENDMETHOD demo_cond. Pasta Cui+C 366 Coty Cui+C Open Chi+C 367 Open Others See Cui+C See Cui+C 368 Coty Cui+C See Cui+C 369 METHOD demo_cond. Pasta Cui+C 370 DATA(result) Coty Coty Cui+C 360 Seare Link for Selection Seare Link for Selection	350						
358 Rever File type int4 359 Swe Cut-5 360 Open ABAP Type Hierarchy Cut-1 361 Swe Cut-5 362 Open ABAP Type Hierarchy Cut-1 363 Out->write(data Swe Cut-7 364 Out->write(name Swe Cut-7 366 ENDMETHOD. Open With Open With 367 Open With Open With Open With 368 Copy Cut-1 Cut-2 369 METHOD demo_cond. Paste Cut-1 369 METHOD demo_cond. Paste Cut-1 360 Copy Cut-1 Cut-2 360 Cospy Cut-1 Cut-1 361 Cospy Cut-1 Cut-1 363 Cospy Cut-1 Cut-1 366 ENDUCE, Reduction Operator diation Sinter File Cut-2 Cut-2 367 Taggle Active and Inactive Version Cut-2 Toggle Comment Cut-2 368 Coverage As Cut-1 Sinte File Ad	<u>*</u> 357	DATA(result) = RE	ø	Undo Typing	Ctrl+Z	TYPE tt_string	
359 Size Ctri-S Ine IN tasks 360 Gen ABAP type Hearchy Ctri-T F4 361 Open ABAP type Hearchy Ctri-T F4 362 Navigate To Base Base Ctri-S Fa 363 Out ->write (data Swee Ctri-S Fa 364 Out ->write (data Swee Ctri-S Fa 365 Out ->write (name Swee Ctri-S Fa 366 ENDMETHOD. Ctri-S Fa 367 Open Others Swee Ctri-S Fa 368 Open Others Swee Ctri-S Fa 369 METHOD demo_cond. Fa Charge Package Assignment Ctri-C 360 Ctri-S Reame. Atr-Shift-S Fa 360 Swee Cade Assignment Ctri-C Ctri-S Fa 360 Consige Package Assignment Ctri-C Ctri-S Fa Switz Fir Sware Cade Assignment Ctri-C Ctri-S Fa Switz Fir Sware Link for Selection Sware Link for Selection Format Shift-FI Swre Link for Selection Sware Link for S	358			Revert File		l type int4	
360 360 361 361 362 Gata ABAP Type Hearchy Cdri-Type 363 Out ->write(data a Open in Project Ctri+Att-P Open With Open With Open With Open Others Fill alue #(base r (line-summ open Project) 366 ENDMETHOD, Show in Alt-Shift-W > Open ABAP Type Hearchy Cdri-X 367 Gata Open Mon Open Others Show in Alt-Shift-W > Open Others Show in Alt-Shift-W > Open Others 367 Gata Open ADATA (result) = CC Cdri C Ctri-Xit-C Corey Ctri-C Corey Ctri-C 368 Gata Open Mon Open Cond. Pate Ctri-Xit-C Corey Ctri-C Corey Ctri-C 369 METHOD demo_cond. Pate Ctri-Xit-C Corey Ctri-C Corey Ctri-C 360 Core Code Alt-Shift-W > Open Files Ctri-Shift-File Christian Ctri-Shift-File 360 Source Code Alt-Shift-Shift-W > Open ABAP Decrement Ctri-Shift-File Christian Ctri-Shift-File 360 Syntax Share Link for Selection Format Block Ctri-Shift-File 361 Shift-File Corerage As Corerage As Corerage As 362 Shift-File Corerage As Corerage As Shift-File	359			Save	Ctrl+S	ine IN tasks	
361 362 362 363 363 out->write(data 365 out->write(name 366 ENDMETHOD. 367 Open NRh 368 Open NRh 369 METHOD demo_cond. 370 DATA(result) = CC Goid Class [lass-relevant Local Types [local Types]local Types]local Types [local Types]local Types [local Types]local Types [local Types]local Types]local Types [local Types]local Types [local Types]local Types [local Types]local Types]local Types [local Types]local Types [local Types]local Types [local Types]local Types [local Types]local Types]local Types [local Types]local Types [local Types]local Types]local Types]local Types [local Types]local Types]local Types]local Types [local Types]local Types]local Types]local Types]local Types [local Types]local Types]local Types]local Types]local Types [local Types]local Types]loc	360		°e.	Open ABAP Type Hierarchy	F4		
Joint Solution So	361			Quick Type Hierarchy	Ctrl+T	L 1	
364 out->write(data 365 out->write(name 366 ENDMETHOD. 367 Open Others 368 Open Others 369 METHOD demo_cond. 370 DATA(result) = CC Get Whre-used List Ctrl-Shift-G 369 METHOD demo_cond. 370 DATA(result) = CC Get Whre-used List Ctrl-Shift-G 360 Cost *Gobal Class-cleavent Local Types [tec:] Famame *Gobal Class-cleavent Local Types [tec:] Famame *Gobal Class-cleavent Local Types [tec:] Famame *Gobal Class [class-relevant Local Types [tec:] Famame *Gobal Class [class-relevant Local Types [tec:] *Coreage As *Coreage As Chrl-Striction *Coreage As Coverage As *Debug As Profile As *Coreage As Coreage As *Coreage	262			Navigate To	F3		
36.3 364 out->write(data 365 out->write(name 366 ENDMETHOD. Open With 0pen With 367 368 369 Open With 0uick Outline Ctrl+Alt+P> 0pen With 0uick Outline > 367 368 369 METHOD demo_cond. 370 DATA(result) = CC 0py Ctrl Ctrl+X Corpy Ctrl+X Ctrl+Shift-C Rename > 369 METHOD demo_cond. 370 Paste Ctrl+X Ctrl+Shift-C Rename = '20220929' THEN VALUE #(Ctrl-Shift-C Rename ** Source Code Alt+Shift-SP Coverage As Coverage As Ctrl+Shift-C Rename Alt-Shift-SP Coverage As ** Source Code Alt+Shift-SP Coverage As Ctrl+Shift-F Coverage As Togle Comment Ctrl-Y Add Comment ** Source Code Alt+Shift-SP Coverage As Open Althesis Shift-FI Delete unused variables (all) Alt+U ** Source Code Alt+Shift-SP Coverage As Openet Method Ctrl-Shift-FI Delete unused variables (all) Alt+U ** Bebug As Openet Mith Ctrl+Alt+Co Replace With Shift-FI Delete unused variables (all) Alt+U ** Compare With Corr aspic With Ctrl+Alt+Co Replace With Show Method Information Alt+Shift-R Show Method Information ** A non-generic diat type stypes. Thift File Withide	362			Navigate to Declaration	Shift+F3	aiue #(base r (.	line-summa
364 out->write(data out->write(name 365 Open Whh > 366 ENDMETHOD. Open Others > 367 Quick Outine Ctrl-O 367 Quick Outine Ctrl-O 369 METHOD demo_cond. Pasta Ctrl-C 370 DATA(result) = CC Get Wher-used List Ctrl-Shift-G 700bal Class Gass-relevant Local Types [Local Types]et Ctrl Change Package Assignment Quick Fix Cuick Fix Ctrl-Shift-G Rename Alt-Shift-S5 REDUCE, Reduction Operator Gil Syntax Source Code Alt-Shift-S5 Vistor trippeting = transport Organizer Value & Ctrl-V Ctrl-V Syntax Source Code Alt-Shift-S5 Vistor trippeting = transport (rate with for Selection Ctrl-V Vistor trippeting = transport (rate with for Selection Coverage As Coverage As Vistor trippeting = transport (rate with for Selection Coverage As Coverage As Vistor trippeting = transport (rate with for Selection Coverage As Coverage As Vistor trippeting = transport (rate with for Selection Coverage As Coverage As <t< td=""><td>363</td><td></td><td></td><td>Open in Project</td><td>Ctrl+Alt+P ></td><td></td><td></td></t<>	363			Open in Project	Ctrl+Alt+P >		
365 out->write(name 366 ENDMETHOD. 367 Show In 368 Cuick Outline 369 METHOD demo_cond. 370 DATA(result) = CC ************************************	364	out->write(data		Open With	>		
366 ENDMETHOD. 367 Galaxy Cutine 368 Cut 369= METHOD demo_cond. 369= METHOD demo_cond. 370 DATA(result) = CC Stow in a data sign and inclus (construction operator construction construct	365	out->write(name		Open Others	>).	
367 368 369 METHOD demo_cond. 370 DATA(result) = CC © Get Where-used List Crt+X Class Class-relevant Local Types [Local Types [L	366	ENDMETHOD.		Show In	Alt+Shift+W >	1	
367 Cut Cut Cut+X 369 METHOD demo_cond. Paste Cut+V 370 DATA(result) = CC Get Where-used List Cut+Sift+G *Global Class Class-relevant Local Types [Local Types [Loc	367			Quick Outline	Ctrl+O		
368 Copy Ctrl+C 369 METHOD demo_cond. Paste Ctrl+C 370 DATA(result) = CC Ede Where-used List Ctrl+Shift+R Ede Where-used List Ctrl+Shift+R *Global Class Class-relevant Local Types Local Types Local Types Local Types Test Ctrl Change Package Assignment Alt+Shift+S Alt+Shift+S *Problems Properties Transport Organizer Test Change Package Assignment Curl+F3 ABAP - Help > ABAP - Programming Language > Program Flow Local *Activate Ctrl+F3 Toggle Comment Ctrl+ Syntax ************************************	260		Å	Cut	Ctrl+X		
3699 METHOD demo_cond. 370 DATA(result) = CC Get Where-used List Ctrl+Shift+G Rename Alt+Shift+R *Global Class Class-relevant Local Types Local Types Test Clip Change Package Assignment Cuick Fix Ctrl+Shift+S Problems Properties Transport Organizer Test ABAPHelp > ABAP Programming Language > Program Flow Local Ctrl+Shift+S REDUCE, Reduction Operator Coll Syntax Toggle Active and Inactive Version With fix transport Organizer Coverage As Coverage As Coverage As	368			Сору	Ctrl+C		
370 DATA(result) = CC Get Where-used List Ctrl+Shift+G = '20220929' THEN VALUE #(*Global Class Class-relevant Local Types Local Types Local Types I to Class Change Package Assignment Quick Fix Ctrl+Shift+G *Problems Properties I Transport Organizer Togle Comment Ctrl+7 ABAP - Hele > ABAP - Programming Language > Program Flow Local * Activate Ctrl+F3 REDUCE, Reduction Operator * Activate Ctrl+F3 Syntax Share Link for Selection Format * Share Link for Selection * Share Link for Selection Format Block * Coverage As Coverage As Coverage As * Debug As * Debug As * Debug As * Team * Coverage With Ctrl+Alt+C+ * Compare With Ctrl+Alt+C+ Show Chalancement Options * Validate Yalidate Yalidate Show ABAP Language Help	369⊜	METHOD demo_cond.		Paste	Ctrl+V		
Rename Alt-Shift+R *Global Class (class-relevant Local Types Local Types [Local Types [Loca	370	DATA(result) = CC	臣	Get Where-used List	Ctrl+Shift+G	= '20220929' THEN	VALUE #(
**Global Class Class-relevant Local Types Local Types Local Types Text CL Change Package Assignment Quick Fix Ctrl+1 Problems Properties Transport Organizer Text ABAP - Help > ABAP - Programming Language > Program Flow Local Syntax Activate Ctrl+F3 MEEDUCE type((Late carge) TINT (x1 = rhs1) (x1) < x1 > extracting Assignment (x2 = rhs2) (x2) > exrespr1) (x1) < x2 > exrespr2) (x2) > exrespr2) (x2) < x2 = exrespr2) (x2) > exrespr2) (x2) > exrespr2) (x2) < x2 = exrespr2) (x2) > exrespr2) (x2) < x2 = exrespr2) (x2) > exrespr2)		<		Rename	Alt+Shift+R		
Problems Properties Transport Organizer Te ABAP - Heig > ABAP - Programming Language > Program Flow Looit Image: Activate Ctrl+Ti ABAP - Heig > ABAP - Programming Language > Program Flow Looit Image: Activate Ctrl+Fi REDUCE, Reduction Operator Image: Activate Ctrl+U Syntax Unlock Ctrl+U Image: Activate Coverage As Image: Activate Image: Activate Activate Coverage As Image: Activate Activate Image: Activate Activate Coverage As Image: Activate Activate Image: Activate Activat	* <u>G</u> lobal Clas	ss <u>C</u> lass-relevant Local Types <u>L</u> ocal Types <u>T</u> est Cla	₿.	Change Package Assignment			
ABAP - Help > ABAP - Programming Language > Program Flow Logic Source Code Alt+Shift+S > Ioggle Comment Ctrl + / ABAP - Help > ABAP - Programming Language > Program Flow Logic Mathematic Ctrl + F3 Add Comment Ctrl + <	Problem	ns 🔲 Properties 具 Transport Organizer 📔 Te		Quick Fix	Ctrl+1		
ABAP - Help > ABAP - Programming Language > Program Flow Logit Activate Ctrl+F3 Add Comment Add Comment Ctrl+ REDUCE, Reduction Operator Import ABAP Doc from Descriptions Import ABAP Doc from Descriptions Import ABAP Doc from Descriptions Syntax Share Link for Selection Coverage As Coverage As Import ABAP Doc from Descriptions INITY (x1 = rha1) (x1) × x1 > rm (x2 = rha2) (x2) × x2 =				Source Code	Alt+Shift+S>	loggle Comment	Ctrl+7
REDUCE, Reduction Operator Image: Syntax I	ABAP - Help	ABAP - Programming Language > Program Flow Logic	淅	Activate	Ctrl+F3	Remove Comment	Ctrl+>
Syntax Important operations REDUCE type((let exp) INIT (x1 = rhs1) (x1) = wrexpr1) (x1 xx) = mrexpr2) (x2 x2) = wrexpr2) (x2 x2) = wrexpr2) (x2 x2) = mrexpr2) (x2 x2) = mrexpr	REDUC	CE, Reduction Operator 🏼 🕼	23	Toggle Active and Inactive Version		Import ABAP Doc from Descriptions	Curry
Final Share Link for Selection Format Shart FF IMIT f(at exp) Share Link for Selection Format Shart FF IMIT f(at exp) f(at exp) Share Link for Selection Format Shart FF IMIT f(at exp) f(at exp) Share Link for Selection Format Shart FF IMIT f(at exp) f(at exp) Share Link for Selection Format Shart FF IMIT f(at exp) f(xi = rhs)/((xi) > = wrexpr1)/(xi) < f(xi) > f(xi) > f(xi) < f(xi) > f(xi	Syntax		6	Unlock	Ctrl+U		0.10.51
Image: Second	BEDUC	T time (//	Share Link for Selection		Format Format Plack	Shift+FI
IMI XI = rink] XI > memory XI XI > memory XI XI > memory XI XI > memory XI XI > memory XI XI > memory XI XI > memory XI XI > memory XI XI > memory XI XI > memory XI XI > memory XI XI > memory XI XI > memory XI XI > memory XI XI > memory XI XI > memory XI XI > memory XI XI > memory XI XI > memory XI XI > memory XI XI > memory XI XI > memory XI XI > memory XI XI > memory XI XI > memory XI XI > memory XI > memory XI XI > memory XI XI > memory XI XI > memory XI >	[<u>le</u>	et exp] []] []] (st = shet)] ((st) = susses t)] (st (st)]	0	Coverage As	>	Pormat Block	Ctri+Shift+FI
FOR for exp1 FOR for exp2 NEXT (x1 = += -= *= /= 66= rha1/ (<x1> = += (x2 = += -= *= /= 66= rha2/ (<x2> = +=)</x2></x1>	INI	$ \frac{ \mathbf{x} = rns1}{ \mathbf{x} = rns2} \frac{ \mathbf{x} + rns1}{ \mathbf{x} + \mathbf{x}$		Coverage As	>	Delete unused variables (all)	Alt+Shift+U
Image: Constructor expression with the reduction operator repression with the reductin operator repressi	FOR	for expl	0	Run As	>	Extract Method	Alt+Shift+M
NEXT Image:	POR	tor expz	*	Debug As	>	Rename	Alt+Shift+R
(x2 = i+=i=r=i/=i/kk= rhs2/i (<x2> = i+=i Team Contraction operation F2 Effect Compare With Ctrl+Alt+C > Show Code Element Information F2 A constructor expression with the reduction operator REDUCE Replace With Ctrl+Alt+C > Show Method Information Alt+F2 • A non-generic data type dtype. Validate Update Navigation Index Update Navigation Index • The # character as a symbol for the operand type. Show ABAP Language Help F1</x2>	NEX	(x1 = += -= *= /= 66 = rha1) (<x1> = += </x1>		Profile As	>	Occurrences in File	Ctrl+Shift+11
Effect Replace With Ctrl+Alt+C > Show Code Element Information Tz A constructor expression with the reduction operator REDUCE Validate Show Method Information Alt+F2 • A non-generic data type dtype. Validate Update Navigation Index • The # character as a symbol for the operand type. Show ABAP Language Help F1		x2 = += -= *= /= && = rhs2 (<x2> = +=)</x2>		Team	>	Show Code Element Information	E2
Effect Replace With Show Enhancement Options A constructor expression with the reduction operator REDUCE Validate Show Enhancement Options • A non-generic data type dtype. Preferences Show ABAP Language Help F1				Compare With	Ctrl+Alt+C >	Show Method Information	Alt+F2
A constructor expression with the reduction operator REDUCE Validate View Validate View Validate View Validate View View View View View View View Vie	Effect			Replace With	>	Show Enhancement Options	
A non-generic data type dtype. Preferences Show ABAP Language Help F1 The # character as a symbol for the operand type.	A construc	tor expression with the reduction operator REDUCE		validate		Update Navigation Index	-
The # character as a symbol for the operand type.	• A n	ion-generic data type atype.		Preferences		Show ABAP Language Help	F1
	• The	e # character as a symbol for the operand type.					

Figure 72 Calling the ABAP Language Help from the Context Menu

3.2.5.15 The ABAP Language Help View

The corresponding documentation is displayed as an HTML document in the ABAP Language Help View. This allows forward navigation via hyperlinks.



Image: Proplus Image: Program Image: Program Flow Logic > Iteration Expressions >	
REDUCE, Reduction Operator	^
Syntax	
<pre> REDUCE type([let exp] INIT (x1 = rhs1) (<x1> = wrexpr1) (x1 <x1> TYPE dtype1) (x2 = rhs2) (<x2> = wrexpr2) (x2 <x2> TYPE dtype2) FOR for exp1 FOR for exp1 NEXT (x1 = += -= *= /= &&= rhs1) (<x1> = += -= *= /= &&= wrexpr1) (x2 = += -= *= /= &&= wrexpr2))) </x1></x2></x2></x1></x1></pre>	
Effect A constructor expression with the reduction operator REDUCE creates a result of a data type specified using type from one or more iteration expressions. The following can be specified for type: A non-generic data type atype.	
 The # character as a symbol for the <u>operand type</u>. If the data type required in an operand position is unique and known completely, this type is used. 	~

Figure 73 ABAP Language Help View

As in many views in Eclipse, there are some useful standard buttons here.



Figure 74 Button Bar of the View

- The yellow double arrow links the view to the active editor. This means that the view always displays the appropriate help for a statement on which the cursor is currently located.
- The green pin holds the content of the view. If help is requested again with F1, a new view for ABAP Language Help opens.
- The yellow arrows to the right and left are used for navigation (analogous to a web browser).
- With the help of the printer symbol, you can create a paper-like copy of ABAP Help with the appropriate hardware.
- The two symbols A with the arrows pointing up and down are responsible for changing the font size.
- The search field can be used to search the complete ABAP Help, including other languages such as CDS or BDL.



3.2.5.16 Application Help

In addition to the ABAP Keyword Documentation (or ABAP Language Help), SAP provides so-called Eclipse Help plug-ins for each development scenario. To do this, click

$\textbf{Help} \rightarrow \textbf{Help Contents}$

in the menu bar to open the Help Browser.



Figure 75 Navigation to Help Content

You can recognize the SAP Help plug-ins by the yellow book icon. Currently, the following Help plug-ins are available:



Working with ADT

🔟 Help - ABAP Development Tools
Search: Go Scope: All topics
Contents 👜 - 🚿 - 🖻 🔄 🗖
🗄 🇇 Eclipse Platform User Guide
🗄 🧇 Java development user guide
🗄 🌑 Plug-in Development Environment Guide
🗄 🍄 SAP (On-Premise) - ABAP Programming Model for SAP Fiori
🗄 😤 SAP (On-Premise) - ABAP Web Services
🗄 🐤 SAP (On-Premise) - BOPF Developer Guide
SAP (On-Premise) - Web Dynpro ABAP Development User Guide
SAP - ABAP CDS Development User Guide
* SAP - ABAP Development Tools Release Notes
SAP - ABAP Development User Guide
About the ABAP Development User Guide
E Getting Started
E III Tacks
Ims and Tricks (ABAP Core Tools)
■ III Security Guide
🗉 💷 What's New in ABAP Core Development
🗄 🧇 SAP - ABAP for SAP HANA Development User Guide
🗉 🧇 SAP - ABAP RESTful Application Programming Model
🗄 🧇 SAP - BOPF Development User Guide

Figure 76 Overview of Available Help and Documentation

With the help of the search (Search) you can search for keywords. You can use Scope to limit the search to one or more help plug-ins.



K Help - ABAP Development Tools	
Constru	×
Search: Scope: All topics Show all topi	
Contents	
Eclipse Platform User Guide	×
Java development user guide	
	e
🗷 🔶 SAP (On-Premise) - ABAP Programming Model for SAP Fior	de
🗷 🔶 SAP (On-Premise) - ABAP Web Services	onment Guide
🗷 🗢 SAP (On-Premise) - BOPF Developer Guide	rogramming M
🗷 🔶 SAP (On-Premise) - Web Dynpro ABAP Development User G	Neb Services
SAP - ABAP CDS Development User Guide)eveloper Guide
SAP - ABAP Development Tools Release Notes	ynpro ABAP Dev
SAP ABAP Development Ligge Guide	ient User Guide
About the ABAP Development User Guide Bab About the ABAP Development User Guide	ools Release No
Gotting Started Getting Started Getting Started	Jser Guide
Rev., Eq. (B) SAP - ADAP IOI SAP IOI SAP - ADAP IOI SAP - ADAP IOI SAP ION	ation Programm
Taske	Jser Guide
Reference	opment User Gu
Tins and Tirks (ABAP Core Tools) If an and Tirks (ABAP Core Tools)	lipse-based SAF
Security Guide	
🖲 🖼 What's New in ABAP Core Development	
B 🔶 SAP - BOPF Development User Guide	>
SAP - Support Guide for Eclipse-based SAP Development Tools	Cancel

Figure 77 Search in Help

In wizards that offer the ? icon, you can open the context-sensitive help. This will take you directly to the respective help content that is available for the wizard.

🔟 New ABAP Class	— 🗆 X
ABAP Class (i) Specify a value for field 'Project'	Contents 🌮 Search
Project: * Package: * Add to favorite packages Name: * Description: * Original Language:	About ABAP Class Development objects are the individual parts that are used to build an ABAP application. See also: ABAP Development Objects Transport Request
Superclass: Interfaces: Add Remove	 Creating ABAP Classes Creating ABAP Exception Classes More results: Search for ABAP Class Search for New ABAP Class
Sack Next > Finish Cancel	

Figure 78 Further Help and Documentation



You can click so-called Active Links in a help page to open an Eclipse UI from the help. You can recognize Active Links by the green icon with the white arrow.

SAP - ABAP Development User Guide > Tasks > Using Troubleshooting Tools > Debugging ABAP Code

Setting ABAP Debugging Preferences

Context

You can change how the ABAP debugger behaves with the General Settings.

Procedure

- 1. Open the Odebugger preference page. (Choose INVindow > Preferences > ABAP Development > Debug.)
- 2. Change debugger settings as required. The settings apply to all of your ABAP projects.

Enable debugging of system programs

Mark this checkbox to make ABAP system pi this option is not activated, then the debugge

Figure 79 Display of Navigation

The help content that is identical to that which is available to you in ADT Client is also available online on the SAP Help Portal.

3.2.6 Refactoring Code with ADT

As briefly mentioned in Chapter 2 - Motivation for ADT, the numerous functions in ADT also allow code to be refactored. But what is refactoring, what advantages does it offer and what functions are provided by ADT? These questions will be answered in detail in the following section.

Refactoring refers to the modification of source code in order to improve its structure and readability without changing the functions. This means that no new functionalities are added, no functionalities are eliminated and the correctness of the solution is maintained, i.e. it continues to deliver the correct result. No new bugs will be introduced.

Maintaining correctness is certainly the most important aspect and one that is most difficult to achieve in the SAP world. The easiest way to prove correctness is through automated testing. Unfortunately, these are not very common in the SAP cosmos, have been little supported in the past, and are often difficult to implement because the structures of historical ABAP code are difficult to implement. Thus, however, the creation of automated testability is a primary goal of refactorings.



In addition, there are other reasons for refactoring:

- Increasing the comprehensibility of the code ("clean code")
- Improve the customizability of code for extensions
- Reduction of technical debt
- Updates of deprecated commands/modules

Refactoring is an integral part of software development and should be done during daily development in order to maintain a certain standard of quality. It is not advisable to carry out special "refactoring sprints" or similar, as these are often viewed skeptically by financiers or not approved at all. The authors therefore recommend observing the Boy Scout rule: "Always leave the code better than you found it."

In the past, this was usually associated with a great deal of effort with the SE80. Due to the lack of support for the IDE, the refactorings had to be carried out manually to a large extent. This high manual effort and their susceptibility to errors led to a low acceptance of this process and clean code in general.

With ADT, this situation has now changed. If there are no automated tests as a double bottom, it is still possible to carry out so-called save refactorings, which we would like to describe here. A save refactoring is characterized by the fact that it is carried out with the help of tools, i.e. by functions of the IDE or with additional plug-ins. This eliminates the risk of introducing new errors into the code due to manual changes. Automation makes it easy to carry out refactorings and thus become part of daily work.

ADT offer the following refactorings via Quick Assists (CTRL+1):

- 1. Rename Identifier renaming within a code block or globally
- 2. Extract Method Extract a method from source code or from an expression
- 3. Extract Constants Convert text literals into constants
- 4. Extract Variables Extract and convert variables
- 5. Move Member Modify and move attributes of classes
- 6. Exception Handling Automated creation/transformation of exception blocks

In particular, the Rename and Extract Method functions support the developer in keeping the code clean and avoiding or reducing code redundancies.

For example, since the rename function handles identifiers not only within the unit, but across all users, it is now easy to change an inappropriately chosen name to a name that better fits the overall context. There is no risk that users will be forgotten and errors will be built into the code.



The Extract function analyzes the selected code, provides assistance with parameter assignment and replaces the location of the code with the call of the newly created method. If the method to be extracted has a comment, it is used as a suggestion for naming the method.

Furthermore, the authors recommend the use of the plug-in "ABAP Quick Fix" (https://marketplace.eclipse.org/content/abap-quick-fix) by Lukasz Pegiel, which is described in Chapter 7 - Plug-ins. In general, the refactoring tools in ADT provided by means of quick fixes offer enormous help both in the creation and revision of existing code. The use of this helpful plug-in improves the currently edited code on the one hand, but also helps to apply the newer language constructs yourself when creating new code, if you are not yet experienced in them.

3.2.7 Versioning and Comparison

The context menu item "Compare with" hides some of the most important features for daily work. These work for all source code editors in ADT, not just for the development of ABAP code.

9	Unlock	Ctrl+U		Local History	
//	Share Link for Selection			Each Other	N
Q.	Coverage As	>		Revision History	
	Coverage As	>		S4H 100 brandeisi en	
0	Run As	>		TRL EN	Alt+Shift+C
夺	Debug As	>		TRL EN 1	
	Profile As	>		TRL_EN_2	
	Compare with	Ctrl+Alt+C >	*	New Project	Ctrl+Alt+Shift+P



3.2.7.1 Local History – The local Visioning

The Local History provides access to older versions of the current source code document from the Eclipse Workspace that the user is currently working with. Each time the object is saved, a version is drawn. This means that you can easily track your own work over the course of hours and days and easily switch back to older versions.

Since the local version history only refers to your own Eclipse workspace, it can happen that you also have changes on another device or that a colleague has last made changes.



3.2.7.2 Revision History – Version Management of ABAP

Under the menu item Revision History, you can access the "normal" source code management of the ABAP server, which was already available in the SAP GUI. By default, versions are pulled here when a transport request is released.

The versions of the revision history are accordingly globally accessible to all users, regardless of the workspace of the Eclipse installation.

3.2.7.3 Other Project Name – Compare across systems

If you pick a project from the list, you can compare across systems. This can be a system from the same system landscape, e.g. the production system, or a completely different system. This means that code from different systems that do not have an RFC connection to each other can be compared from ADT. In complex system landscapes, this offers a great advantage over GUI-based comparison options.

3.2.7.4 Comparison View

In the Comparsion View you can see the current state of the source code on the left and the version selected for comparison on the right. The discrepancies are highlighted:

- 1. Green are the things that are new in the current version
- 2. Red are the things that have been deleted and are in
- 3. The changes are highlighted in grey

With the buttons, the end old state can be restored by copying from right to left. However, it is also possible to make changes directly in this comparison view on the left side. After saving, the comparison is repeated.



E ⁰ [TRL] ZBC_TASKS ×	- D
III Database Table Compare 🖇	M 🖾 🖘 🌆 🖓 🖓
	Local history: ZBC_TASKS - 30.10.2022, 14:45:19
1@EndUserText.label : 'Tasks'	1@EndUserText.label : 'Tasks'
2@AbapCatalog.enhancement.category : #NOT_EXTENSIBL	2@AbapCatalog.enhancement.category : #NOT_EXTENS
3@AbapCatalog.tableCategory : #TRANSPARENT	3@AbapCatalog.tableCategory : #TRANSPARENT
4@AbapCatalog.deliveryClass : #A	4@AbapCatalog.deliveryClass : #A
5@AbapCatalog.dataMaintenance : #RESTRICTED	5@AbapCatalog.dataMaintenance : #RESTRICTED
<pre>6define table zbc_tasks {</pre>	6 define table zbc_tasks {
7	7
<pre>8 key client : abap.clnt not null;</pre>	8 key client : abap.clnt not null;
<pre>9 key task_key : zbc_task_key not null;</pre>	<pre>9 key task_id : zbc_task_id not null;</pre>
10 summary : zbc_task_summary;	10 task_key : zbc_task_key;
11 status : zbc_task_status;	11 summary : zbc_task_summary;
12 project : zbc_project_key;	12 status : zbc_task_status;
13 description : abap.char(1000);	13 project : zbc_project_1d;
14 assignee : abp_creation_user;	14 description : abap.char(1000);
15 type : zbc_task_type;	15 assignee : abp_creation_user;
16 author : zbc_author;	16 type : zbc_task_type;
<pre>1/ changed_at : abp_locinst_lastchange_tstmpl;</pre>	1/ author : zbc_author;
18 created_at : abp_creation_tstmp1;	18 changed_at : abp_locinst_lastchange_tstmpl;
19 due_date : zbc_due_date;	19 created_at : abp_creation_tstmp1;
20 Solution : 20C_Solution;	20 due_date : 20C_due_date;
21 priority : ZDC_priority;	21 SOLUCION : ZDC_SOLUCION;
22 product : 20c_product_10;	22 priority : 20c_priority;
23 new_rieid : abap.accp;	25 product : 20C_product_10;
	25
235	2.5 5

Figure 81 Comparison View – Comparison of two Versions

If you want to take over an old version completely, you can select the appropriate version directly from the context menu using Replace With \rightarrow Local History.

Compare with Validate	Ctrl+Alt+C >	
Replace With	>	Previous from Local History
Preferences		Local History

Figure 82 Context Menu for Completely Transferring a Version from Local Version Management

3.2.8 Documentation with ABAP Doc

3.2.8.1 What is ABAP Doc?

ABAP Doc enables code-based documentation, such as methods and their parameters.





Figure 83 ABAP Doc Documentation of the Method

ABAP Doc is a feature that is only supported in ADT. Instead of the form-based editor with the option of briefly describing methods, which is no longer available in ADT, a much more powerful replacement has been established with the ABAP Docs, which is also available in a similar form in other programming languages (e.g. JavaDoc).

In the following, the use of ABAP Docs in the context of classes/methods is discussed in detail for better readability of the text. However, the ABAP Docs can also be applied to other development artifacts, such as function modules (see SAP Help).

The ABAP Doc function can be used to enter textual descriptions for classes and their methods. In addition, descriptions can also be stored in ABAP Doc for the individual parameters and exceptions.



The notes to be created with ABAP Doc are created in the definition area. However, the added value arises above all from the simple access to this documentation by the user. This is possible both at the call point and in the area of implementation of development artifacts using the F2 key. In addition, the texts created in ABAP Docs can even be formatted using HTML tags. In this way, the documentation can be enriched with headings or text formatting and thus presented in an even more appealing and structured way.

The "Synchronized" tag is used to transfer descriptions created in ABAP Doc to the short texts displayed in the SAP GUI, so that the headings are visible even when viewed using SE24/SE80.

This can be useful if objects contain enhancements that are not edited directly in ADT and therefore the modification still has to be done in the GUI-based tools.

We do not recommend a mixture of ABAP Doc and GUI short texts. The use of ABAP Doc is the method of choice to relieve the functional code of comments and to provide the user with helpful information about the development artifacts.

3.2.8.2 Use of Quick Fixes to Create ABAP Doc

Creating the ABAP Doc is easy by calling the Quick Fixes. To do this, select the method definition, call the Quick Fixes, and choose "Add ABAP Doc". If a method definition has been changed, for example, by adding a parameter, and an update of the documentation is required, the ABAP Doc can be updated by selecting the ABAP Doc area directly instead of the method signature and thus calling the Quick Fixes.

3.2.8.3 Further Information about ABAP Doc

Further information about ABAP Doc can be found in the official SAP Help (for example) under ABAP Doc - ABAP Keyword Documentation (sap.com) (7.50), in the example class CL_DEMO_ABAP_DOC and in the User Guide.

3.2.9 Executing Source Code

Even in ADT, open source code can still be executed comfortably. F8 is used to initialize an SAP GUI instance of the respective system and execute the open development object. In the case of classes, for example, this corresponds to the "Test Class X" function, and in the case of reports, the report is executed normally.



	[ISH] ZCL_CALC_DSP	×		
Menü 🖌 📀	× « .	🔇 🚫 🔇) 🖶 🖥	
Testen Klasse ZCL_CA	LC_DSP			
🔺 🛅 📔 🏓 Behandi	ler			
TestObject->				
Groß/Kleinschreibung aktiv	v 🔲			
<pre>ZCL_CALC_DSP</pre>	{0:13*\C	LASS=ZCL_CALC	C_DSP}	
Interfaces				
Attribute				
CALCULATE		Ð		
Figure 84 Executing a class	s in SAP GUI			
# [TRL] ZDSP 🛛 [ISH] ZPRG	_ADT_TEST_REPORT	🗾 [ISH] ZPRG	ADT_TEST_REPO	$RT \times$
Menü 🖌 💽	~ 《 🖪	🔇 🔕 🕄	🖶 H H	1
Test Report				
()				
Do something	۲ ل			

Figure 85 Result of Execution

If any development object is to be executed, **ALT+F8** can be used to search for an object via the Object Finder. Any project can be selected, i.e. an SAP system connected to ADT. This does not have to be the system in which development is currently taking place – it can also be a quality assurance system. It is important that the ADT functions are activated for this system or that the corresponding authorizations exist.



Enter search string and /or filter criteria (see help for details)		
zprg*	Project Selection - X	
<u>M</u> atching items:	<u>C</u> hoose a project:	
 ZPRG_BASAR_PROT_ANALYSE_KZ (Program) Programm ZPRG_BAS_MN_TREE (Program) Programm ZPRG_BDWAL_PORTAL_ANALYSE (Program) Programm ZPRG_BDWAL_PORTAL_LZA_CALC (Program) Programm ZPRG_BDWAR_PORTAL_LZA_VGL (Program) Programm ZPRG_BDWAR_PORTAL_ANALYSE (Program) Programm ZPRG_BDWAR_PORTAL_ANALYSE (Program) Programm ZPRG_BDWAR_PORTAL_ANALYSE_BT (Program) Programm ZPRG_BDWAR_PORTAL_ANALYSE_KZ (Program) Programm ZPRG_BDWAR_PROT_ANALYSE_KZ (Program) Programm ZPRG_BDWAR_PROT_ANALYSE_KZ (Program) Programm ZPRG_BDW_MN_TREE (Program) Programm ZPRG_BEDAL_PORTAL_ANALYSE (Program) Programm ZPRG_BEDAL_PORTAL_LZA_CALC (Program) Programm ZPRG_BEDAL_PORTAL_LZA_VGL (Program) Programm ZPRG_BEDAL_PORTAL_LANALYSE (Incogram) Programm 	type filter text ISH_100_dpanzer_de [ISH, 100, DPANZER, DE] S43_010_dpanzer_de [S43, 010, DPANZER, DE] TRL_EN [TRL, 100, CB9980000948, EN]	
ZPRG_BEDAR_PORTAL_ANALYSE_BT (Program) Programm More than 50 results	New OK Cancel	

Figure 86 Selection of the Project

The menu item "Run \rightarrow Run History" provides additional information about objects that have already been executed, so that their execution can be conveniently repeated.

Classes that implement the interface if_oo_adt_classrun can also be executed directly via F9 as a console application and thus generate output in the console.

Reports that generate write output can be executed via F9. The WRITE output is then redirected to the console as well.



```
▶ G ZCL_CLASS_RUN_DEMO > ● IF_OO_ADT_CLASSRUN~MAIN
  1 CLASS zcl class run demo DEFINITION
  2
       PUBLIC
  3
       FINAL
  4
       CREATE PUBLIC .
  5
       PUBLIC SECTION.
  6
  7
  8
         INTERFACES if_oo_adt_classrun .
  9
       PROTECTED SECTION.
 10
       PRIVATE SECTION.
 11 ENDCLASS.
 12
 13
 14
 150 CLASS zcl class run demo IMPLEMENTATION.
 16
 17
18⊝
       METHOD if oo adt classrun~main.
 19
         out->write( 'Text auf der Console').
 20
       ENDMETHOD.
 21 ENDCLASS.
Global Class Class-relevant Local Types Local Types Test Classes Macros
🔲 Properties 🛛 🔚 Templates 🛄 Bookmarks 🔊 Feed Reader 🛄 Transport Organi:
ABAP Console
Text auf der Console
```

Figure 87 Output to the Console

3.2.10 Data Preview

The View Data Preview can be used to display data from database tables and (CDS) views. The view opens either by selecting a corresponding object in the Project Explorer and pressing the shortcut $\mathbf{F8}$, or by using the context menu.



Project Explorer 🐹 🍳 CDS Navigator				
	\$ 7	7 8		
✓		^		
> 🖶 /DMO/FLIGHT_DRAFT (28) Flight Reference	Scen	aric		
✓	nce Sc	enc		
> Core Data Services (1)				
✓ i Dictionary (120)				
✓				
> III /DMO/AGENCY Flight Reference S		- New Detabase Table		
> III /DMO/AIRPORT Flight Reference S		New Database lable		
> III /DMO/BOOK_SUPPL Flight Reference	Ľ	New Data Definition		
> III /DMO/BOOKING Flight Reference	∎\$	New Append Structure		
> III /DMO/CARRIER Flight Reference S	P	Duplicate		
> III /DMO/CONNECTION Flight Refer				
> III /DMO/CUSTOMER Flight Reference		Open		
> III /DMO/FLIGHT Flight Reference Sci		Open in Project	Ctrl+Alt+P >	
> III /DMO/SUPPL_TEXT Flight Referen	12	Show in ABAP Repository Tree	>	
> III /DMO/SUPPLEMENT Flight Refere		Open With	>	Data Preview
> ## /DMO/TRAVEL Flight Reference Sc		- F		Distingant log
> 😰 Data Elements (43)	Ð	Сору	Ctrl+C	
> 🗊 Domains (30)	×	Delete	Delete	N SAP GUI
> T Lock Objects (1)	\$	Add to Favorite Objects		
> The Tructures (19)				
> Table lypes (15)	€.	Get Where-used List	Ctrl+Shift+G	
> 🖭 Views (1)				

Figure 88 Start the Data Preview from the Table

The shortcut **F8** also works if you have a corresponding object open and the focus is on the object.

When opened, the view immediately executes the data selection and then lists the selected data in table form. In addition, it shows the number of selected rows and the time required for them.

🛗 Raw Data						C.	🔹 🚺 Show Log	Max. Rows: 100 🚔 🍕
Filter pattern	🍸 Filter pattern 🖉 🤻 100 rows retrieved - 9 ms (partial result) 🚓 SQL Console Data Aging n Number of Entries 🖩 Select Columns 🦄 Add filter 💌 🗐 💌							
RB CLIENT	R TRAVEL_ID	AGENCY_ID	CUSTOMER_ID	BEGIN_DATE	B END_DATE	12 BOOKING_FEE	12 TOTAL_PRICE	CURRENCY_CODE
200	00004141	070002	000001	2021-06-24	2021-07-01	500.00	500.00	EUR
200	00004142	070001	000001	2021-06-24	2021-07-01	0.00	0.00	
200	0000003	070046	000093	2021-02-27	2021-12-26	80.00	4164.00	USD
200	0000004	070042	000665	2021-02-27	2021-12-26	40.00	1871.00	USD
200	0000005	070007	000225	2021-02-27	2021-02-27	20.00	992.00	LISD

Figure 89 Display of the Data Preview

You have various options in the view to adjust the selection. These include:

- Number of selected rows
- Selected columns
- Filter
- Sorting (clicking on the column header)



In addition, you can search for a pattern in the displayed data (incl. ? and * as a joker sign). Matching data is then highlighted in color and bold. In addition, the total number of affected entries and a log of actions executed can be viewed. The Save button allows you to save the displayed values in different formats within a file. It is even possible to generate an ABAP value statement, which can be very useful for creating test data.

In the case of CDS views with associations, it is possible to follow the associations and thus display the linked data. To do this, select one of the data records and select the desired association using the arrow at the top.

/DMO/I_CONNEC	TION_R >				
Raw Data		List of Associations			
▼ Filter pattern ∅	20 row	ੀਡ _Airline → /DN ੀਡ _Flight → /DM	/IO/I_Carrier [0 1] IO/I_Flight_R [1 *]		ng
AirlinelD AB	Connection				Arri
SQ 0001			To follow the associ	ation, choose an association from t	the list 50:00
SQ 0002	2	SIN	SFO	06:30:00 vorm.	09:15:00

Figure 90 Navigation via Associations

A highlight of the Data Preview View is the SQL Console. Based on the selected columns, specified filters and the collation, an SQL Select Statement is generated, which is then used to select the data.

	Check 💽 Run 👻 Data Aging Max. Rows: 100 😜	🔢 Raw Data					
1	SELECT // // // // // // // // // // // // //	Y Filter pattern	🖉 🂐 4 rows retrieved	- 24 ms		72 N	lumber of Entries 📗 🝷
3	/DMO/I_CONNECTION_R~CONNECTIONID,	AIRLINEID	CONNECTIONID	DEPARTUREAIRPORT	DESTINATIONAIRPORT	DEPARTURETIME	ARRIVALTIME
5	/DMO/I CONNECTION R~DESTINATIONAIRPORT,	LH	0400	FRA	JFK	10:10:00 vorm.	11:34:00 vorm.
6	/DMO/I_CONNECTION_R~DEPARTURETIME,	LH	0401	JFK	FRA	06:30:00 nachm.	07:45:00 vorm.
7	/DMO/I_CONNECTION_R~ARRIVALTIME	LH	0402	FRA	EWR	01:30:00 nachm.	03:35:00 nachm.
8	FROM	LH	0403	EWR	FRA	06:09:00 nachm.	07:30:00 vorm.
10	WHERE						
11	· AIRLINEID = 'LH'						
11	· AIRLINEID = 'LH'						

Figure 91 SQL Console

This SQL statement can be customized, checked and executed. The following rules and restrictions apply:

- Only SELECT statements according to ABAP Open SQL syntax are allowed.
- It is possible to create aggregations and complex selections, e.g. with JOIN and UNION.



- Only read-only accesses are possible (no SQL statements with data changes)
- Keywords related to internal tables cannot be used

You can also call the SQL Console directly by opening the context menu for the ABAP project in the Project Explorer and selecting the SQL Console menu item. The view displays the most recently used SQL statement and executes it immediately.

All in all, the SQL Console is a powerful tool that can be used to easily execute selections, evaluate data or test adjustments to selects.

Details about the Data Preview can be found in the Eclipse Help for ADT.

3.2.11 Core Data Services

In the area of Core Data Services (CDS), there are different file types that can be created in ADT:

- Data Definitions (DDLS) Source code files for data models in the following variants:
 - DDIC Based CDS Views
 - o CDS View Entities
 - Abstract CDS entities
 - Hierarchies
 - Extensions of the views
 - CDS Table Functions
- Access Control Files (DCLS) Access Definitions
- Metadata Extensions (DDLX) Outsourcing of annotations from the CDS definition
- Behavior Definitions (BDEF) Behavior Definitions for RAP Business Objects
- ✓ G Core Data Services (4)
 - - > 🛃 ZDCLS_DEMO_ADT Data Control File
 - Behavior Definitions (1)
 - > B ZCDS_DEMO_ADT Demo CDS View Entity
 - - > D ZCDS_DEMO_ADT Demo CDS View Entity
 - - E ZCDS_DEMO_ADT Demo MetaData Extension

Figure 92 Object Types of the CDS in Navigation



The source code editors for the different file types of Core Data Services behave largely like the source code editor for ABAP code. Among other things, the following features are included:

- Code Completion (CTRL+SPACE) Default values that fit in context.
- Element Info (F2) Information about the element on which the cursor is placed.
- Pretty Printer (SHIFT+F1)

One difference is the colors used in the editor.

A basic problem with Core Data Services is that the properties of an object (e.g. a CDS View Entity) are composed of several files and the properties of the data sources. These files each declare their affiliation, and the data sources propagate their field properties (annotations). This is very practical for the expansion concept. But transparency suffers because the files are not necessarily in the same development package.



Figure 93 Different Files Define the Properties of a CDS View Entity



In order to get a complete picture of a CDS object, taking into account all files and propagations, tools are therefore necessary. These include:

- Element Info
- Dependency Analyzer
- Active Annotations

3.2.11.1 Item Info for CDS

With **F2** or the separate Element Info View, you get a good overview of the data structure and associations for a CDS View, regardless of where they were defined. It also displays all the relevant extension files.

zcds_demo_adt (Root Vi Demo CDS View Entity	ew)			^		
Client Handling ✓ Client dependent ✓ Client session variabl	e used					
Generated Objects © SEARCH.SEARCHABLE	SEARCH Domain @S	Search				
Extended with D ZCDS DEMO ADT EX						
Metadata extended with	1		И			
Column	Component Type	Data Type	Description			
🐖 Userld	zbc_user_id	char(12)	User			
Firstname	zbc_firstname	char(50)	Fist name			
Lastname	zbc_lastname	char(50)	Last name			
Email Gondor	zbc_email zbc_email	char(50)	E-Mail Address			
DateOfBirth	zbc_genuer zbc_date of birth	dats(8)	Date of Birth			
TheNumber	2bc_ddtc_of_birtin	numc(2)	bate of birth			
zzNewExtendedField	l i i i i i i i i i i i i i i i i i i i	numc(2)				
Accordiation Target	Cardina	l:+.,				
TasksToDo 20	<u>DS TASKS</u> 0*	uty		V		
<			>			
@ <> <> <-> <-> <-> <-> <-> <-> <-> <-> <]			.:;		

Figure 94 Overview of a CDS View Entity Using Element Info



3.2.11.2 Dependency Analyzer

The Dependency Analyzer provides a good overview of the origin of the data. It is accessed via the context menu.

1		Open With	>	di	Data Preview
		Show In	ک Alt+Shift+W >	æ	Activation Graph
	Þ	Сору	Ctrl+C	F	Dictionary Log
	×	Delete	Delete	a	Active Annotations
	☆	Add to Favorite Object	5	0	Annotation Propagation
	€	Get Where-Used List	Ctrl+Shift+G	÷	Dependency Analyzer
	-		etti onne o	Act	ivation of worklist (CB99

Figure 95 Calling the Dependency Analyzer via the Context Menu

The Dependency Analyzer has three tabs that display information about a view:

The **SQL Dependency Tree** displays the hierarchical structure in tabular form

"Se [TRL] ZC_STATUS ×					
SQL Dependency Tree				🔗 🖽	🖻 🕹 察
SQL Name	SQL Relation	Object Type	Entity Name	Database Object	Access Contro
V T ZC_STATUS		CDS Projection View (STOB)	zc_status	True	None
✓ III ZI_STATUS	From	CDS View Entity (STOB)	zi_status	True	None
ZBC_STATUST	From	Database Table (TABL)		True	
✓ III ZI_STATUS_TEXT	Left Outer Join	CDS View Entity (STOB)	zi_status_text	True	None
ZBC_STATUS_TEXT	From	Database Table (TABL)		True	
Find				🕂 Next	🔶 Previous
SOL Dependency Tree SOL Depen	ndency Granh Comn	levity Metrics			

Dependency Tree SQL Dependency Graph Complexity Metrics

Figure 96 SQL Dependency Tree

The **SQL Dependency Graph** displays the same information graphically.





Figure 97 SQL Dependency Graph

• The **Complexity Metrics** tab displays additional information on the overall complexity of the CDS view, including all source views.



🖫 [TRL] ZC_STATUS 🗙 🔃 [T	RL] ZC_STATUS_	TEXT	
Complexity Metrics			
Used Data Sources			
Database tables:	2		
Database views:	2		
CDS views with parameter:	0		
CDS table functions:	0		
SQL Operations			
JOIN operations:	1		
SET operations:	0		
GROUP BY clauses:	0		
Performance Related Function	on Calls and Ope	erations	
Function calls:	0		
Explicit type CAST operations:	0		
CASE expressions:	0		
SQL Dependency Tree SQL Dep	endency Graph	Complexity Metrics	

Figure 98 Complexity Metrics

3.2.11.3 Active Annotations

The Active Annotations view is also accessed via the context menu in the navigation. It displays the values of all active annotations of the views:

- At the view level , only exactly the annotations that have been defined in the view are visible.
- At **the field level**, all valid annotations are visible. Their origin, such as data element, metadata extension or data source, is also displayed.
- At parameter level

Some properties, e.g. field texts, can already be defined by the data elements and then propagated to the view. This is exemplified in the following figure:



Working with ADT

R Probl.	🔲 Prope 📔 Templ.	🔃 Book 🔊 I	Feed 📮 Tran	s 📃 Cons 📋 InfoP	BW-R (@ Active × □ □
					🔗 🕀 🖂 🖇
Active	Annotations for En	tity zi tasks			
type filter	text	,			
		A	T 1. 1T .	0	0 D. T
Annotate	d Elements	Annotation Value	Iranslated lext	Origin Data Source	Origin Data Type
∽ 🔊 zi_t	asks				
∽ @	Entity annotations				
~	@AccessControl				
	authorizationCheck	#NOT_REQUIRED			
~	@EndUserText				
	label	'Tasks'			
~	TaskKey				
~	@EndUserText				
	heading			ZBC_TASKS (Database Table)	ZBC_TASK_KEY (Data Element)
	label		Key	ZBC_TASKS (Database Table)	ZBC_TASK_KEY (Data Element)
	quickInfo		Key	ZBC_TASKS (Database Table)	ZBC_TASK_KEY (Data Element)
~	@ObjectModel				
	upperCase	true		ZBC_TASKS (Database Table)	ZBC_TASK_KEY (Data Element)
~	Summary				
~	@EndUserText				
	heading			ZBC_TASKS (Database Table)	ZBC_TASK_SUMMARY (Data Elem
	label			ZBC_TASKS (Database Table)	ZBC_TASK_SUMMARY (Data Elem
	quickInfo		Task summary	ZBC_TASKS (Database Table)	ZBC_TASK_SUMMARY (Data Elem

Figure 99 Active Annotations of a View



4 <u>Troubleshooting Tools in Eclipse</u>

This chapter provides an overview of the tools that are available for troubleshooting in the ABAP Development Tools . This includes, but is not limited to, debugging development artifacts, performance analysis, and other analysis and debugging methods.

In addition to the description of the tools, we provide tips for their sensible use. For detailed functional descriptions, we recommend the official documentation and, if available, provide helpful links.

4.1 Troubleshooting with the ABAP Development Tools

Anyone who has managed to get started with the ABAP development tools and appreciates the advantages of ADT in development is reluctant to continue developing in an SAP GUI-based environment. But the ABAP development tools are not just an environment for creating and modifying code. ADT contains numerous tools for the general analysis of code, the finding of errors and the detailed analysis of the performance of the functionalities. In summary, these activities can be referred to as troubleshooting.

But as with the entry into development with ADT, there is a steep learning curve to overcome with the troubleshooting tools, the conquest of which is rewarded with numerous advantages. As an example, it should be mentioned here that it is possible to change the faulty code directly in the debugger instead of cumbersomely opening a new mode in parallel, searching for the relevant location and then making the change. There are many advantages to discover in the troubleshooting tools, which are described here in the individual sections.

In addition to the direct advantages, the aspect of "Eclipse ADT as the same tool for everything" should not be neglected in the area of troubleshooting.

For example, if you are well versed in the SE80 debugger, it will be more difficult to make the switch, as the advantages of the ADT debugger must first be worked out. For the development process as a whole, however, it makes sense to use one tool for all use cases and to reduce or completely avoid switching back and forth between environments.



4.2 The Debugger in ABAP Development Tools

In Eclipse, a debugger is available for analysis purposes. This is started via the button

automatically changes to the perspective of the debugger.

4.2.1 Breakpoints and Soft-Breakpoints

Breakpoints in Eclipse are exclusively external breakpoints. The breakpoints are drawn each time the software is run. The process is interrupted at the appropriate point.

Breakpoints can be set in all perspectives in the editor to the left of the line number. Alternatively, breakpoints can be set via the context menu. Set breakpoints are indicated by a blue dot next to the line of code.

In addition to the normal breakpoints, the ABAP development tools offer the possibility of soft breakpoints. These are set via the context menu and marked with a green dot. In contrast to the standard breakpoints, the program flow is only stopped at this point if the software is running in the debugging context. Otherwise, soft breakpoints are skipped.

Debug 🗙 🔚 Project Explorer 🚜 Servers) 🛛 🖪 🧏 🖇 🖤	🗉) 🕝 [C15] ZCL_AM_VI 🛛 🚱 [C14] ZCL_MCM_A 🛛 😰 [C14] SFLIGHT_D 🗙 🔯 [C14] SFLIGHT_D 📄 "16		🝽= Variables 🗙 🍫 Breakpoints 👯 Expression					
ABAP Debugger [C14, 010, TC53 (Thomas Foehn), DE]	> Q SELIGHT DATA GEN > commit > loop			20				
✓	37 DELETE FROM sticket CLIENT SPECIFIED WHERE mandt = sy-mandt.	^	Name	Value				
✓	38⊖IF (sy-subrc <> 0) AND (sy-subrc <> 4).		A Contervariable					
SFLIGHT_DATA_GEN start-of-selection [event]	39 MESSAGE 1610 WITH 'STICKET'.			000000000000000000000000000000000000000				
	40 ENDIF. 41		A TAB STICKETTI	[384x9(158)]Standard				
	42 DELETE FROM snvoice CLIENT SPECIFIED WHERE mandt = sy-mandt.		SY-SUBRC	0				
	$43 \ominus$ IF (sy-subrc $\leftrightarrow 0$) AND (sy-subrc $\leftrightarrow 4$).		> @ Globals					
	44 MESSAGE 1610 NITH SMVOLCE .							
	46							
	47 COMMIT WORK.	1. Sec. 1.						
	50 * fill the database table STICKET							
	51 *							
	52 53 CONSTANTS, exterior and separat TVDE i VALUE EAA			N				
	53 CONSTANTS ENTRES DE CONSTRUCTOR I VALUE 300.	-		60				
	55 WITH HEADER LINE INITIAL SIZE 9000,							
	56 entry_cnt TYPE i VALUE 0.							
	57							
	59 REFRESH itab_sticket.							
	60 SELECT mandt carrid connid fldate bookid customid							
	61 FRUM SDOK							
	♦ 64⊖LOOF AT itab_sticket.		<					
	65 ADD 1 TO entry_cnt.		000000000000000000000000000000000000000					
	67 itab sticket-place = airport.	66 Itab sticket-ticket = original. 67 itab sticket-place = airport						
	68 INSERT INTO sticket VALUES itab_sticket.							
	69⊖ IF entry_cnt < 20.							
	70 - More than I ticket for some of the first 20 bookings 71 itab sticket = copy.							
	72 itab_sticket-place = gate2.							
	73 INSERT INTO sticket VALUES itab_sticket.							
	74 itab sticket = stuardess. 75 itab sticket-place = gate3							
	76 INSERT INTO sticket VALUES itab sticket.	75 Itab_sticket-place = gate3. 76 INSERT INTO sticket VALUES itab sticket.						
	77 ENDIF.	~						
	<	>						
	📮 Console 😰 Problems 🕖 Debug Shell 🚮 ABAP Unit 🌄 ABAP Internal Table (Debugger) 🗙			o 🗘 🗞 🛃 🕴 🖓				
	Table Data Columns ITAB_STICKET[] × (384x9(158))Standard Table							
	Filter pattern			LG				
	Row MANDT CARRID CONNID FLDATE BOOKID CUSTOMID TICKET PLACE ARCHIVE							
	1 010 0.0 0017 20050200 00000000 00000000 00000000							
	2 010 44 0017 20051116 0000000 00000317 0000							
	3 010 44 0054 20050311 00000000 00003476 0000							
	Δ 010 ΔΔ (054 2050408 0000000 00000341 19000							

4.2.2 Debugging Perspective

Figure 100 Debugging Perspective in Eclipse


The debugging perspective in Eclipse provides a quick overview of the program code, call stack, variable contents, and contents of internal tables. The variables and internal tables can be selected by double-clicking in the program code. They are displayed on the right side.

xpressions 🛛 🗖 🗍
Value
0
{0:33*\CLASS-POOL=ZCL

Figure 101 Values of the Variables in the Debugging Perspective

4.2.3 Special Behavior in the Debugger

In the debugger of the ABAP Development Tools, it is possible to modify and activate the code directly. However, in the current debugging context, the coding is not yet active. The program run must be restarted.

4.2.4 Additional Information

Further information about debugging with the ABAP Development Tools in Eclipse can be found in the SAP blogs. The following two should be emphasized. They describe cases and solutions that can occur when working with the debugger.

https://blogs.sap.com/2020/04/21/adt-abap-debugger-what-to-do-if-your-program-does-not-stop-at-breakpoints/

https://blogs.sap.com/2015/11/02/breakpoint-validity-scope-and-activation-conflictsin-abap-development-tools-adt/



4.3 Checkpoint IDs and dynamic logpoints

A very helpful tool in the field of error analysis and debugging are the so-called checkpoint IDs. These can be created via transaction SAAB or in ADT under "others". These IDs are defined using the following commands:

- BREAK POINT ID [GROUP NAME]
- LOG POINT ID [GROUP NAME]
- ASSERT ID [GROUP NAME]

anchored in the code. For the detailed syntax and options of the commands, please refer to SAP Help.

As in the debugger of the SAP GUI, these dynamic breakpoints can be activated for debugging or used for logging. The main advantage here is that the developer can add breakpoints to important places in the code in advance. If the code is to be analyzed, the Checkpoint ID only needs to be activated once. When the unit is called, the debugger is called at the appropriate point if the breakpoint is active.

In order to use the checkpoints effectively, it is advisable to create appropriate templates, which can then be easily called up using quick fixes (see Chapter 3 - Working with ADT in the section on templates).

While checkpoint IDs can also be used in the GUI-based debugger, ADT for onpremise systems also offers the possibility to set dynamic log points in the debugger, which can be used to read internal program values. This option is useful if it is not possible to change the productive code or if an analysis has to be carried out very promptly on the production system.





Figure 102 Creation of a Log Point via the Context Menu



Create Logpoint						×
nter logpoint properti	es					
Enter one or more varia	ibles to be logg	ged.				
-		-				
▼ Location						
Main Program /NEO	/CL_	_DPC_EXT (Globa	l Class)			
Program Line /NEO	/CL_	_DPC_EXT (Globa	I Class) / Lin	e 1135		
- Activity						
Log Simple Variable V	alues					~
Log Simple Variable V	alues					
SQL Trace for Current	Statement					
Table Buffer Trace for	Current Statem	nent				
User-Defined Logging						~
	L					
 Description 						
 Activation 						
Status		Active				~
Active for User			Browse	"*" for ar	ny User	
Active on Server		All Servers				~
Active Until		08.02.2023 🗐 🔻	19:21:35 🌲			
Days to Keep Before D	eletion	7 Deleted	l at 15.02.202	3, 19:21:3	5	
Max. Log Events for In	ternal Sercion	100				
-	Lethal Dession					
	ternar Session	100				
	ternar bession					
	ternal Session					

Figure 103 Attributes when Creating a Log Point

In the dialog, you can decide what should be recorded in the log, you can give the dynamic logpoint a description that will then be used in the log output, and you can specify various criteria for whether (based on a condition – hidden in the screenshot – and/or user/server) and how long the log output should take place. Created logpoints



are displayed in the editor on the left margin and on the right margin next to the vertical scroll bar and listed in the "Logpoints" view:

11340 Hettood /isbep/if may core sry_runtime-init. 1135 super-Xisbep/if may core sry_runtime-init(1136 io_context = io_context 1137 iv_maespace = iv_maespace 1138 iv_service_document_name = iv_service_docume 1139 iv_version = iv_version 1140	nt_nam	ie [Active Logpo	int: /NEO/CL_	DPC_EXT in	iit [Key [iv_servic	e_document_	name, iv_vers	ion], Created By	: Change At 07.02.2023	19:31:34 By)
🖹 Pro., 🛄 Pro., 🔄 Tem., 🛄 Boo., 🗟 Fee., 📮 Tran., 🌠 ATC	🔽	ABA 📋 ABA.	🗐 ABA	🛷 Sear 🚸 at	va 🎋 Deb 🏌	g ABA 📃 Co	m 🧲 ATC	📚 ABA.	. 😪 aba 🕞 ABA 💈) ABA 📄 ABA 🕨	Log X
Project / Main Program / Location	Lo	Last Executio	Current S	Active/Inactive	Last Change	Changed	Created By	User Filter	Description	Deactivation At	^
EN [Not logged on yet]											
v 🛃 100 en [user filter refreshed:											
V O /NEO/CL_ E_DPC_EXT											
/NEO/CL	<u>0</u>		Active	07.02.23, 19:31	07.02.23, 19:31				/NEO/CL	08.02.23, 19:21	

Figure 104 Log Points View in the Debugging Perspective

We recommend that you refer to the SAP documentation (On-Premise), to know details about the application. This blog entry also offers a good introduction to this: Dynamic Logpoints in ABAP | SAP-Blogs

4.4 **Performance Analysis**

For an integrated and graphical performance analysis, ADT offer you convenient access to the ABAP Profiler (on-premise/cloud) as a successor to transaction SAT, among others. There are several ways to start the profiler:

If you have opened an executable program (on-premise only), a console application (ABAP Cloud only), or an artifact with assigned unit tests, you can start the profiler directly from the "Profile as" context menu. Alternatively, you can also use the wizard, which can be accessed via the Run menu.

In case you need a different starting point, you can use trace requests to create requests for the start of the profiler. To do this, first display the "ABAP Trace Requests" view (which can be found in the view list below ABAP, see Views and Perspectives in Chapter 3 - Working with ADT). In this view, after selecting a system with the Create-Trace-Request-Icon, you get a wizard. This allows you to select different triggers for the start of the profiler. If the system is accessed directly with HTTP(S) accesses (note: this is usually not the case in a hub/FES configuration in the backend), a pattern can be used for the URL, e.g. the name of the OData service with a leading and trailing asterisk. Furthermore, an RFC call of a function module or the start of a background job and various other triggers can be used to start the trace. In SAP Fiori development, the function module /IWBEP/FM_MGW_HANDLE_REQUEST can be used as a trigger in a hub/FES configuration for OData accesses in the back-end system. The number of trigger activations can be limited, as well as there is a possibility for time



limitation. You can use various settings to control the scope of data collection. Trace requests can be deleted via the context menu in the list.

									چ 🐻
type filter text									
ABAP Trace Requests	Description	Time	Date	Object	Executions	Expires at	Object Type	User	CI ^
v 🛃en [user filter:), refreshed: 20:10:53, 21.11.2022]									
	SKD01_CORE_SRV	19:58:11	21.11.2022	*SKD01_CORE_SRV*	1/3	20:58 21.1	URI Pattern	1	10
and a second sec	/IWBEP/FM_MGW_HANDLE_R	19:55:33	21.11.2022	/IWBEP/FM_MGW_HANDLE_R	0/3	20:55 21.1	Function Module		10

Figure 105 ABAP Trace Requests in the Debugging Perspective

In addition, there is the possibility to start the trace from the ADT debugger (onpremise/cloud).

In the case of trace requests, the number of traces already generated per request can be updated via the update icon in the trace request view. Via the context menu or by double-clicking on it, you can jump to the view with the list of traces. From there, the display of a trace can be opened. You can either use the context menu to jump to the various tabs of the analysis or to the overview page by double-clicking.



Figure 106 Context Menu of a Trace



In addition to a brief overview of the runtime, the overview page also offers direct jumping points into the various tabs.

Overviev	N		
▼ <u>G</u> eneral	Information	▼ Analysis <u>T</u> ools	
<u>T</u> itle:	*SKD01_CORE_SRV*	Condensed Hit List	Shows top consumers aggregated to called units (methods, forms, etc.)
<u>I</u> D:	_00,AT000000	Hit List	Shows top consumers by call position
D <u>a</u> te: Senver:	21.11.2022, 19:58:16	: <u>Aggregated Call Tree</u>	Shows aggregated call stacks in a hierarchy
<u>U</u> ser:		🚪 Aggregated Timeline	Shows diagram of aggregated call stacks and time consumed
Aggregatio	n: No Aggregation (Call Sequence available)	ᄩ <u>Call Sequence</u>	Shows Sequence of trace events in a hierarchy
<u>S</u> ize:	507 KB	📒 Call Timeline	Shows diagram of trace events and time consumed
		Database Accesses	Shows database accesses
▼ Runtime	e Distribution		
<u>Overall:</u>	0,24s		449/ ABAD
ABAP:	0,11s (44%)		56% Database
Data <u>b</u> ase:	0,14s (56%)		0% System
System:	< 0,01s (0%)		

Figure 107 Overview of the Properties of a Trace

In particular, the graphical analysis of the timeline is only available in ADT and facilitates analysis. If you move the mouse pointer over the blocks, you will be shown details directly and can also navigate directly to the source code via the context menu.



Overview Condensed Hit List Hit List Aggregated Call Tree Aggregated Timeline Call Sequence Call Timeline Database Accesses

Figure 108 Aggregated Overview of a Trace History



The SQL trace, which provides the PLV files for a visual analysis of the query plans, especially in the case of HANA as a database, can be started in the context menu of a system in the Project Explorer (also works in the ABAP environment). However, the results are then displayed in a Web application outside ADT (or in transaction ST05).



Figure 109 Jump to the SQL Trace

For visual analysis, you can currently also install the SAP HANA administration tools from the HANA Studio in ADT Eclipse (or the complete SAP HANA Studio in parallel) and thus configure the automatic start of the visual analysis from the transaction. To do this, set the user parameter HDB_OPEN_STUDIO to X





Figure 110 Getting started with managing User Parameters

<	Maintain U	Jser Profile
	Password TOTP Registration M	Exit
Address Defaults Pa	rameters	
Parameters		
		\square
SET/GET Parameter ID	Parameter value	Short Description
HDB_OPEN_STUDIO	X	Open HANA Studio or SQL Analyzer when a Planviz is created

Figure 111 Setting the User Parameter HDB_OPEN_STUDIO

and associate opening *.plv files with the right eclipse .exe in the operating system. For example, on Windows, there is the option "Open with..." in the context menu of the file in File Explorer and there you will find the option "Find another app on this PC" at the bottom, after clicking you have to search for and select the "eclipse.exe" of your Eclipse/ADT installation as well as the option "Always this app ... use".





Figure 112 Selection of eclipse.exe as a New Way to Display *.plv Files

After selecting a row in the SQL trace list of the ST05 or selecting a trace record and an SQL statement in the SQL trace analysis of the Technical Monitoring Cockpit, you can request the HANA PlanViz Query Plan visualization:

$\left \right>$	Q (*	< ∨ ₿		~ @ Z V V # #~		
-	Start Time	Σ Duration	Σ Records	Program Name	Object Name	Statement
		6.612.223	1.338			
	14:37:18.362	2.752	1	/NEO/CL_SKD_FLAT_CONFIG=====CP	/NEO/SKD_FCONF_S	SELECT <fda read=""> WHERE "MANDT</fda>
	14:37:18.383	2.566	1	/NEO/CL_SKD_FLAT_CONFIG=====CP	/NEO/SKD_FCONF_S	SELECT <fda read=""> WHERE "MANDT</fda>
	14:37:18.388	2.241	1	/NEO/CL_SKD_FLAT_CONFIG=====CP	/NEO/SKD_FCONF_S	SELECT <fda read=""> WHERE "MANDT</fda>
	14:37:18.433	32.451	44	CL_SADL_SQL_EXECUTOR=====CP	/NEO/CSKDNODRESP	SELECT WHERE "MANDT" = '205' AND
	14:37:18.764	2.664	1	/NEO/CL_SKD_FLAT_CONFIG=====CP	/NEO/SKD_FCONF_S	SELECT <fda read=""> WHERE "MANDT</fda>
	14:37:18.788	1.919	1	/NEO/CL_SKD_FLAT_CONFIG=====CP	/NEO/SKD_FCONF_S	SELECT <fda read=""> WHERE "MANDT</fda>
	14:37:18.942	1.766	1	/NEO/CL_SKDEAM_S_PMCSOR_SAMPLECP	/NEO/SKD_INT	SELECT WHERE "MANDT" = '205' AND
	14:37:19.083	2.821	110	/NEO/CL_SKDEAM_S_PMCSOR_SAMPLECP	/NEO/SKD_LEVEL, /NEO/SKD_SKL	SELECT <fda read=""> <join> WHERE</join></fda>
	14:37:19.326	1.540.622	915	CL_SADL_SQL_EXECUTOR======CP	ZI_NEOS_RESOURCETIMESLOTSASS	SELECT WHERE "MANDT" = '205' AND
	14:37:20.877	7.465	39	ZCL_NEOS_RES_TIME_SLOTS=====CP	/NEO/SKD_RES_ADR	SELECT DISTINCT WHERE "MANDT" =
	14:37:20.887	8.449	1	ZCL_NEOS_RES_FOR_SUGGESTION===CP	ZI_NEOS_SKILLMATCH	SELECT <fda read=""> WHERE "MANDT</fda>
	14:37:20.893	2.373	1	ZCL_NEOS_RES_FOR_SUGGESTION===CP	ZI_NEOS_SKILLMATCH	SELECT <fda read=""> WHERE "MANDT</fda>

Figure 113 Selection of a Specific Selection



Troubleshooting Tools in Eclipse

< = SAP Technical Mo	onitoring Coc	kpit \sim						Q	? (i)	<u>д</u> (2)
			SC	L Trace Ar	nalysis					
Trace Directory Trace Records SQI	Statement	Prepared Plan	Executed P	lan						
> 2023-01-30 15:42 🐻 +01:00 (Local	Time)	<	1D		As Collected		~ >		2023-01-3	1 15:42 [🐻
									Dow	nload PLV File
Operator	Subtree Cost	Individual	Exclusive	Rows	Object Sch	Object Na	Table Type	Execution	User CPU	Kernel CP
✓ HEX Search	100.000 %	0.000 %	408 μs	0					0.000 ms	0.000 ms
✓ HEX Search	100.000 %	100.000 %	408 μs	1				HEX	0.244 ms	0.000 ms
✓ Unique Index Lookup "SAPAB	0.000 %	0.000 %	0 µs	0	SAPABAP	REPOLOAD	COLUMN		0.132 ms	0.000 ms
✓ REPOLOAD.\$trexexternalke	0.000 %	0.000 %	0 µs	0					0.132 ms	0.000 ms
UniqueIndexLookupOp	0.000 %	0.000 %	0 µs	0					0.132 ms	0.000 ms
✓ Project	0.000 %	0.000 %	0 µs	0					0.112 ms	0.000 ms
ProjectBufferOp	0.000 %	0.000 %	0 µs	0					0.106 ms	0.000 ms
ProjectFetchOp	0.000 %	0.000 %	0 µs	0					0.006 ms	0.000 ms
> Operator Details										
> Table Details										
> Index Details										

Figure 114 Download the *.plv File

Eclipse then automatically starts the correct view, and the Tab Executed Plan shows the visual analysis of the query plan.



Figure 115 Query Execution Plan Display



For development with the RAP (ABAP RESTful Application Programming Model), there is a special trace tool called Cross Trace (Doku On-Premise/Cloud), which you can use to analyze requests from Fiori apps across the RAP software stack (SAP Gateway, BO Behavior, SADL, ABAP Core). A corresponding separate authorization is required for use.

To start, display the "ABAP Cross Trace" view. This view has two tabs. In the first tab, you can create a new cross-trace configuration in the context menu of a system. A cross-trace configuration can be provided with a description to distinguish it, can be active or inactive, automatic deactivation after a certain number of trace requests is possible, as well as automated deletion at a certain point in time. You can decide whether non-sensitive or sensitive data should be recorded. Optionally, you can filter by user, access type and target (e.g. only a specific OData service; * is also possible here for wildcard filtering), as well as you can specify whether and with which trace level a recording should take place for the respective cross-trace components.



Create cross nace conlige	Jration			
eate Cross Trace Configu	iration in NSC_EN			
Specify the attributes of the	e Cross Trace Configuration and choose whether	the trace is active when the configuration is created	d	
Status				
Activate Trace				
Properties				
Properties	20			
	.es: 20			
Current Number of Traces:	0			
Include Content				
Include potentially sensiti	ve content			
eletion At:	06.02.2023			
Description:	WoRo Test			
Configuration ID:				
lequest Filters				
lser: CB000	0000001		Bro	owse
equest Entry Type:				
equest Entry Name				
Components				
type filter text				
type filter text		ID	Translaud	•
type filter text Component		ID	Trace Level	^
type filter text Component V ABAP BAdls BAdl calls		ID SAP.BC.BADI.CORE	Trace Level	^
type filter text Component ✓ Ø ABAP BAdls Ø BAdl calls ✓ Ø ABAP Behavi		ID SAP.BC.BADI.CORE	Trace Level (3) Expert	^
type filter text Component V ABAP BAdIs BAdI calls V ABAP Behavio Core Runt	pr ime	ID SAP.BC.BADI.CORE SAP.BC.BEHV.CORE	(3) Expert	^
type filter text Component V ABAP BAdIs BAdI calls V ABAP Behavie Core Runt Determina	or ime tions and Validations	ID SAP.BC.BADI.CORE SAP.BC.BEHV.CORE SAP.BC.BEHV.DVM	(3) Expert (3) Expert (3) Expert (3) Expert	^
type filter text Component ✓ ABAP BAdls Ø BAdl calls ✓ ABAP Behavi Ø Core Runt Ø Determina ✓ Managed Run	or ime tions and Validations time (BSP Framework)	ID SAP.BC.BADI.CORE SAP.BC.BEHV.CORE SAP.BC.BEHV.DVM	(3) Expert (3) Expert (3) Expert (3) Expert	^
type filter text Component ✓ ABAP BAdls Ø BAdl calls ✓ ABAP Behavia Ø Core Runt Ø Determina ✓ Managed Run Ø Managed	or ime tions and Validations time (GSP Framework) Runtime (CSP Framework)	ID SAP.BC.BADI.CORE SAP.BC.BEHV.CORE SAP.BC.BEHV.DVM SAP.BC.BSP	(3) Expert (3) Expert (3) Expert (3) Expert (3) Expert	^
type filter text Component	or ime tions and Validations ntime (BSP Framework) Runtime J Runtime (CSP Framework) Runtime	ID SAP.BC.BADI.CORE SAP.BC.BEHV.CORE SAP.BC.BEHV.DVM SAP.BC.BSP SAP.BC.CSP	(3) Expert (3) Expert (3) Expert (3) Expert (3) Expert (3) Expert	^
type filter text Component	or ime tions and Validations ntime (BSP Framework) Runtime d Runtime CSP Framework) Runtime naged Runtime	ID SAP.BC.BADI.CORE SAP.BC.BEHV.CORE SAP.BC.BEHV.DVM SAP.BC.BSP SAP.BC.CSP	(3) Expert (3) Expert (3) Expert (3) Expert (3) Expert (3) Expert	^
type filter text Component	or ime tions and Validations ntime (BSP Framework) Runtime d Runtime (CSP Framework) Runtime naged Runtime Service Provider	ID SAP.BC.BADI.CORE SAP.BC.BEHV.CORE SAP.BC.BEHV.DVM SAP.BC.BSP SAP.BC.CSP SAP.BC.CSP	(3) Expert (3) Expert (3) Expert (3) Expert (3) Expert (3) Expert (3) Expert (3) Expert	^
type filter text Component	or ime itions and Validations ntime (BSP Framework) Runtime d Runtime (CSP Framework) Runtime naged Runtime Service Provider cks	ID SAP.BC.BADI.CORE SAP.BC.BEHV.CORE SAP.BC.BEHV.DVM SAP.BC.BSP SAP.BC.CSP SAP.BC.DSP	Trace Level (3) Expert	^
type filter text Component	or ime itions and Validations ntime (BSP Framework) Runtime d Runtime (CSP Framework) Runtime naged Runtime Service Provider cks ontract Tests o Street Finabling (EEE)	ID SAP.BC.BADI.CORE SAP.BC.BEHV.CORE SAP.BC.BEHV.DVM SAP.BC.BSP SAP.BC.CSP SAP.BC.CSP SAP.BC.CSP SAP.BC.CSP SAP.BC.CSP	Trace Level (3) Expert	
type filter text Component	or ime itions and Validations ntime (BSP Framework) Runtime d Runtime d Runtime naged Runtime Service Provider cks ontract Tests e Event Enabling (EEE)	ID SAP.BC.BADI.CORE SAP.BC.BEHV.CORE SAP.BC.BEHV.DVM SAP.BC.BSP SAP.BC.CSP SAP.BC.CSP SAP.BC.CSP SAP.BC.RAP.ATC.BO_CONTRACT SAP.BC.RAP.SRV.EEE.GENERAL	(3) Expert (3) Expert (3) Expert (3) Expert (3) Expert (3) Expert (3) Expert (3) Expert (3) Expert	^
type filter text Component	or ime itions and Validations ntime (BSP Framework) Runtime d Runtime Maged Runtime Service Provider cks ontract Tests e Event Enabling (EEE)	ID SAP.BC.BADI.CORE SAP.BC.BEHV.CORE SAP.BC.BEHV.DVM SAP.BC.BSP SAP.BC.CSP SAP.BC.CSP SAP.BC.CSP SAP.BC.RAP.ATC.BO_CONTRACT SAP.BC.RAP.SRV.EEE.GENERAL SAP.BC.RAP.SRV.EEE.RUNTIME	(3) Expert (3) Expert	
type filter text Component	or ime itions and Validations ttime (BSP Framework) Runtime d Runtime (CSP Framework) Runtime naged Runtime Service Provider cks ontract Tests ie Event Enabling (EEE) ervice Generation Runtime	ID SAP.BC.BADI.CORE SAP.BC.BEHV.CORE SAP.BC.BEHV.DVM SAP.BC.BSP SAP.BC.CSP SAP.BC.CSP SAP.BC.CSP SAP.BC.RAP.ATC.BO_CONTRACT SAP.BC.RAP.SRV.EEE.GENERAL SAP.BC.RAP.SRV.EEE.RUNTIME	Trace Level (3) Expert	
type filter text Component	or ime itions and Validations ntime (BSP Framework) Runtime d Runtime (CSP Framework) Runtime naged Runtime Service Provider cks ontract Tests ie Event Enabling (EEE) :rvice Generation Runtime ie Binding V2	ID SAP.BC.BADI.CORE SAP.BC.BEHV.CORE SAP.BC.BEHV.DVM SAP.BC.BSP SAP.BC.CSP SAP.BC.CSP SAP.BC.CSP SAP.BC.RAP.ATC.BO_CONTRACT SAP.BC.RAP.SRV.EEE.GENERAL SAP.BC.RAP.SRV.EEE.RUNTIME SAP.BC.RAP.SRV.ODATA	Trace Level (3) Expert	
type filter text Component	or ime tions and Validations ntime (BSP Framework) Runtime d Runtime (CSP Framework) Runtime inaged Runtime Service Provider cks ontract Tests ie Event Enabling (EEE) :rvice Generation Runtime ie Binding V2 ation and Query (SADL Framework)	ID SAP.BC.BADI.CORE SAP.BC.BEHV.CORE SAP.BC.BEHV.DVM SAP.BC.BSP SAP.BC.CSP SAP.BC.CSP SAP.BC.CSP SAP.BC.RAP.ATC.BO_CONTRACT SAP.BC.RAP.SRV.EEE.GENERAL SAP.BC.RAP.SRV.EEE.RUNTIME SAP.BC.RAP.SRV.EEE.RUNTIME	Trace Level (3) Expert	
type filter text Component	or ime itions and Validations ntime (BSP Framework) Runtime d Runtime (CSP Framework) Runtime naged Runtime Service Provider cks ontract Tests ie Event Enabling (EEE) invice Generation Runtime ie Binding V2 ation and Query (SADL Framework) Metadata Queries Cache Compilation	ID SAP.BC.BADI.CORE SAP.BC.BEHV.CORE SAP.BC.BEHV.CORE SAP.BC.BEHV.DVM SAP.BC.BSP SAP.BC.CSP SAP.BC.CSP SAP.BC.CSP SAP.BC.RAP.ATC.BO_CONTRACT SAP.BC.RAP.SRV.EEE.GENERAL SAP.BC.RAP.SRV.EEE.RUNTIME SAP.BC.RAP.SRV.EEE.RUNTIME SAP.BC.RAP.SRV.ODATA SAP.BC.SADL.BASIS_APIS SAP.BC.SADL.COMPUER	Trace Level (3) Expert	
type filter text Component	or ime itions and Validations ntime (BSP Framework) Runtime d Runtime (CSP Framework) Runtime naged Runtime Service Provider cks ontract Tests ie Event Enabling (EEE) ervice Generation Runtime ie Binding V2 ation and Query (SADL Framework) Metadata Queries Cache Compilation ents	ID SAP.BC.BADI.CORE SAP.BC.BEHV.CORE SAP.BC.BEHV.CORE SAP.BC.BEHV.DVM SAP.BC.BSP SAP.BC.CSP SAP.BC.CSP SAP.BC.CSP SAP.BC.RAP.ATC.BO_CONTRACT SAP.BC.RAP.SRV.EEE.GENERAL SAP.BC.RAP.SRV.EEE.RUNTIME SAP.BC.RAP.SRV.EEE.RUNTIME SAP.BC.RAP.SRV.ODATA SAP.BC.SADL.BASIS_APIS SAP.BC.SADL.COMPILER SAP.BC.SADL.GENERAL	Trace Level (3) Expert	
type filter text Component	or ime itions and Validations ntime (BSP Framework) Runtime Runtime Runtime Runtime Service Provider cks potract Tests ise Event Enabling (EEE) ervice Generation Runtime ise Binding V2 ation and Query (SADL Framework) Metadata Queries Cache Compilation ents	ID SAP.BC.BADI.CORE SAP.BC.BEHV.CORE SAP.BC.BEHV.CORE SAP.BC.BEHV.DVM SAP.BC.BSP SAP.BC.CSP SAP.BC.CSP SAP.BC.CSP SAP.BC.RAP.ATC.BO_CONTRACT SAP.BC.RAP.SRV.EEE.GENERAL SAP.BC.RAP.SRV.EEE.RUNTIME SAP.BC.RAP.SRV.ODATA SAP.BC.SADL.BASIS_APIS SAP.BC.SADL.COMPILER SAP.BC.SADL.GENERAL SAP.BC.SADL.GENERAL SAP.BC.SADL.RUNTIME	(3) Expert (3) Expert	
type filter text Component ✓ ABAP BAdls Ø BAdl calls ✓ ABAP Behavin Ø Core Runt Ø Determina ✓ Managed Managed ✓ RAP Managed ✓ RAP Draft Ma RAP Draft ✓ RAP ATC Che RAP BO Cr ✓ RAP ATC Che RAP BO Cr ✓ RAP Cheterpris Ø General Ø RAP OData Sa RAP Sorvic ✓ SAP Gateway	or ime itions and Validations ntime (BSP Framework) Runtime Runtime Runtime Runtime Service Provider cks potract Tests :e Event Enabling (EEE) ervice Generation Runtime :e Binding V2 ation and Query (SADL Framework) Metadata Queries Cache Compilation ents	ID SAP.BC.BADI.CORE SAP.BC.BEHV.CORE SAP.BC.BEHV.CORE SAP.BC.BEHV.DVM SAP.BC.BSP SAP.BC.CSP SAP.BC.CSP SAP.BC.CSP SAP.BC.RAP.ATC.BO_CONTRACT SAP.BC.RAP.SRV.EEE.GENERAL SAP.BC.RAP.SRV.EEE.RUNTIME SAP.BC.RAP.SRV.ODATA SAP.BC.SADL.BASIS_APIS SAP.BC.SADL.COMPILER SAP.BC.SADL.GENERAL SAP.BC.SADL.GENERAL SAP.BC.SADL.RUNTIME	Trace Level (3) Expert (3) Expert	

Figure 116 Creation of ABAP Cross Traces



After confirming with OK, the configuration is displayed in the view, where the current status (active/inactive), the description and the number of remaining accesses to be recorded can also be viewed. In the context menu, a configuration can be edited and activated/deactivated/deleted/updated. A global update is possible in the upper right corner of the view. To display the results, switch to the second tab of the view. Here you can see a trace for each individual access. In the context menu, you can open a trace (and delete it, etc.). The trace then opens in another view. There you can see the path of access through the individual components and filter the displayed lines or start a text search. You can use the offset column to understand the time course of access. Details are displayed for each row in the Properties View.

[NSC] /SAP/OPU/ODATA4/NEO/SKD01_CONFIG_04_SRV/SRVD | 06.02.23, 18:59:01 | CB0000000001 | 02595F0682DE1EDDA9C8FDF090DBAE5C

53 🕀 🖻 🛯 🖉 🐇

type filter text						
Procedure	Processed Objects	Message	Record Properties	Content Size (Component	^
✓ □ OData Provisioning request		POST Request received	URI:/sap/opu/odata4/neo/skd01_	9943	SAP.GW.SERVER	
□ IF_HTTP_LOCATION_EXC_PLUGIN~GET_LOCA	CL_HTTP_LOCATION_EXC_DEFAULT -	BAdi call		157	SAP.BC.BADI.CORE	
L IF_HTTP_LOCATION_EXC_PLUGIN~GET_LOCA	Υ.			74	SAP.BC.BADI.CORE	
> Get MDP For Exposure		ID = '\$SRVD#/NEO/SKD_CONFIG_RESTYPEGRP'		46	SAP.BC.SADL.RUNTIME	
Get MDP For Exposure					SAP.BC.SADL.RUNTIME	
 Initialize Application 	/NEO/SKD01_CONFIG_04_SRV (Servi	Service Group /NEO/SKD01_CONFIG_04_SRV			SAP.GW.SERVER	
□ Initialize V4 DPC		Initialize		18	SAP.BC.SADL.RUNTIME	
L Initialize V4 DPC					SAP.BC.SADL.RUNTIME	
 OData Entity Set Query 		Querying set from '/NEO/C_SKD_RESOURCETYPEGRPALL'		1168	SAP.BC.SADL.RUNTIME	
Read Context Flags		IS_STICKY_SESSION: false		24	SAP.BC.SADL.RUNTIME	
ABQI Construction		ENTITY: '\$SRVD#/NEO/SKD_CONFIG_RESTYPEGRP~/NEO/C_SKD_RESOURCETYPEGRI		75	SAP.BC.SADL.RUNTIME	
✓ □ Query Entity Collection		(see content for query details)		3312	SAP.BC.SADL.RUNTIME	
> SADL Query		Querying /NEO/C_SKD_RESOURCETYPEGRPALL, 8 elements requested, with paging		769	SAP.BC.SADL.RUNTIME	
L SADL Query		Query retrieved 1 row(s)		702	SAP.BC.SADL.RUNTIME	
GET_LOAD		ENTITY: '/NEO/C_SKD_RESOURCETYPEGRPALL'		42	SAP.BC.SADL.RUNTIME	
L GET_LOAD		Result code: OK		986	SAP.BC.SADL.RUNTIME	
 ABQI Construction 		ENTITY: '\$EXP#CDS#/NEO/C_SKD_RESOURCETYPEGRPALL~/NEO/C_SKD_RESOURCE		81	SAP.BC.SADL.RUNTIME	
✓ □ GET_PROPERTIES		ENTITY: /NEO/C_SKD_RESOURCETYPEGRPALL, KEYS: 1		972	SAP.BC.SADL.RUNTIME	
⊢ GET_LOAD		ENTITY: '/NEO/C_SKD_RESOURCETYPEGRPALL'		42	SAP.BC.SADL.RUNTIME	
L GET_LOAD		FEATURES: [TABLE<3x(7)>]		582	SAP.BC.SADL.RUNTIME	
r GET_LOAD		ENTITY: '/NEO/C_SKD_RESOURCETYPEGRPALL'		42	SAP.BC.SADL.RUNTIME	
L GET_LOAD		ASSOCIATIONS: [TABLE<1x(12)>]		339	SAP.BC.SADL.RUNTIME	
 GET_STATIC_FEATURES 		Request for: /NEO/C_SKD_RESOURCETYPEGRPALL (Root: /NEO/C_SKD_RESOURCETY	ENTITY:/NEO/C_SKD_RESOURCE1	92	SAP.BC.BEHV.CORE	
GET_STATIC_FEATURES		Global Result for: /NEO/C_SKD_RESOURCETYPEGRPALL (Root: /NEO/C_SKD_RESOUR	ENTITY:/NEO/C_SKD_RESOURCE1	92	SAP.BC.BEHV.CORE	
 GET_STATIC_FEATURES 		Instance Result for: /NEO/C_SKD_RESOURCETYPEGRPALL (Root: /NEO/C_SKD_RESOL	ENTITY:/NEO/C_SKD_RESOURCE1	92	SAP.BC.BEHV.CORE	
 CALL_GLOBAL_AUTHORIZATION 		Job Triggers (1)	JOB:{o:2228} GAJOB,LOG:{o:2273	76	SAP.BC.BEHV.CORE	
> CALL_HANDLER(Global Authorization	/NEO/C_SKD_RESOURCETYPEGRPAL	Call (Global Authorization) {o:2218} HANDLER_PROJ	ROOT:/NEO/C_SKD_RESOURCETY	150	SAP.BC.BEHV.CORE	
CALL_HANDLER(Global Authorization)	o				SAP.BC.BEHV.CORE	
 CALL_INSTANCE_CONTROLLERS 		Job Triggers (1)	JOB:{o:2182} FJOB, CONTENT: CON	76	SAP.BC.BEHV.CORE	
> CALL_HANDLER(Instance Features)	/NEO/C_SKD_RESOURCETYPEGRPAL	Call (Instance Features) {o:2145} HANDLER_PROJ	ROOT:/NEO/C_SKD_RESOURCETY	277	SAP.BC.BEHV.CORE	
CALL_HANDLER(Instance Features)					SAP.BC.BEHV.CORE	
L GET PROPERTIES				2008	SAP RC SADE REINTIME	~

				8
Record 1				
General		61 ~path_translated_expanded	/sap/opu/odata4/nco/skd01_config_o4_srv/srvd/nco/skd_config_restypegrp/0001/\$batch	~
Content		62 ~gui_header_handler_field	/sap/opu/odata4	
Call Stack		63 HTTP BODY	batch_id=1675706340487-950	
Record Properties			Content-type.appirtation/nttp Content-transfer-encoding:binary	
Trace Header			GET ResourceTypeGrpAllSet?sap-client=100&\$filter=ResourceTypeGroupAllId%20eq%201%20and%20(IsActiveEntity%20eq%20false%20or%20SiblingEntity/ Accept:application/json;odata.metadata=minimal;IEEE754Compatible=true	ı,
	<		> · · · · · · · · · · · · · · · · · · ·	

Figure 117 Detailed View of Operations

You can jump directly to the triggering line of source code, view the call hierarchy, and so on.



4.4.1 Note on HANA Studio and SAP HANA Tools

HANA Studio is only developed to a limited extent by SAP. A future solution for the visual analysis of query plans within ADT or for the ABAP Cloud is still pending. As a solution outside of Eclipse, there is a Visual Studio Code plug-in that can also open *.plv files and display them graphically, cf. SQL Analyzer Extension.

The SAP HANA tools follow a different release cycle than ADT and are therefore often not included in the current or the "latest" update site. For example, in January 2023, the latest version of SAP HANA tools was only available on the https://tools.hana.ondemand.com/2022-09 site.

4.5 Feed Reader

Feeds enable event-related notifications in ADT including access to a list of previous events. The feeds are displayed in a separate view (see Views and Perspectives in Chapter 3 - Working with ADT). These can be found in the list of views below ABAP with the title "Feed Reader". The individual sources are called feeds and for each feed you can set how often it should be updated by the source and whether notification messages should be displayed in Eclipse.

In addition to the ABAP runtime errors/dumps and system messages described in the documentation under "Getting Feeds" (On-Premise/Cloud), the following information in particular can also be displayed in the Feed Reader (depending on the release status of the source system):

- Gateway/OData errors (can be filtered in a variety of ways, including users)
- ATC results (can be filtered in a variety of ways, including users)
- Enterprise event errors (filterable by channel and user)
- BW Job Repository (can be filtered in a variety of ways, including users)
- URI Creation Error (only relevant for ADT developers)

For this purpose, the events in the pull procedure are queried in the background. However, for the background query from SAP systems to work, you must have accessed the desired systems at least once in some form after an Eclipse start (you can also trigger this by clicking on a feed) and have gone through the login procedure.

In addition, any Atom/RSS feeds can also be subscribed to, this can be, for example, the RSS feed for the last blog posts on a day on blogs.sap.com:



SAP	Community	Topics	Groups	Answers	Blogs	s Ever	nts Programs	Resources	What's New	Explo	ore SAP	Q
	Home > Community > Blogs 								Ask a Question	Write a Blog	g Post ····	
	ABAP Development										ŭ	
	Search the S	SAP Commun	iity					Q				
	✓ Following Write a Blog Post RSS	-					5780	Blog Posts	1948 Contributo	ors 284	5 Followers	
type filter text	Following Write a Blog Post RSS tor feed entry		_		·		5780	Blog Posts	1948 Contributo	ors 284! 	5 Followers	A 8
 type filter text Project / Feed	Following Write a Blog Post RSS	Date	Time	Author	Total	Unrea: ^	5780 Monday, November 07, 202 ABAP at SAP TE Just bios contro & BRD 1	Blog Posts	1948 Contributo	ors 2849	5 Followers	Ă 8
type filter text Project / Feed → (19) Na	Following Write a Blog Post RSS	Date 07.11.2022	Time 14:54:46	Author	Total 20	Unreat ^	5780 Monday, November 07, 202 ABAP at SAP Te Latest blog posts for ABAP (Blog Posts	1948 Contributo	ors 284	5 Followers	Ă A S
type filter text Project / Feed > (19) Na > (19) Na	Following Write a Blog Post RSS for feed entry iduery / Feed Entry iduery / Feed Entry iduery / Seed Entry iduery /	Date 07.11.2022 07.11.2022 07.11.2022	Time 14:54:46 14:54:46	Author	Total 20 20	Unreat ^ 19 19	5780 Monday, November 07, 202 ABAP at SAP Te Latest blog posts for ABAP (Hey everyone) We are a HE premier tech confe	Blog Posts 202:25 PM schEd in 2022 Pevelopment Carine To Imost there: the SA	1948 Contributo 2 Crihouteue Djome PF TechEd in 2022 is j pers, IT practioners,	prs 284	5 Followers	Ă 8
type filter text Project / Feec V (19) Na V (19) E	Following Write a Blog Post RSS for feed entry I Query / Feed Entry ative Feeds Latest blog posts for ABAP Development Become a Paid expert for PLATFORM TECHNOLOGIES - Ask an Expert Peer BAP	Date 07.11.2022 07.11.2022 07.11.2022	Time 14:54:46 14:54:46 14:33:38 14:25:23	Author Ervin Szolke Carine Tchout	Total 20 20	Unrea: ^ 19 19 1	5780 Monday, November 07, 202 ABAP at SAP It Lates biog posts or ABAP Hey everyonel We are a THE premier tach confe Hey everyonel We are a	Blog Posts 20225 PM schEd in 2022 Avereispinent Carine T imost there: the SA rence for all develop who are interested	1948 Contributo 2 Generateue Djome P Techtef in 2022 is pers, IT practitioners; IT graduing a delaifed	just around the c understanding of	5 Followers	A S
type filter text Project / Feed ✓	Following Write a Blog Post RSS tfor feed entry IQuery / Feed Entry Tables blog posts for ABAP Development Jeases blog posts for ABAP Development Backman a Paid expert for PLATFORM TECHNOLOGIES - Ask an Expert Peer BABA Read Feed entry ABAP # SAP Teched in 2022 THY (711 2022 MC2423 - Canae Techedron Demon	Date 07.11.2022 07.11.2022 07.11.2022 07.11.2022 07.11.2022	Time 14:54:46 14:54:46 14:33:38 14:25:23 08:41:09	Author Ervin Szolke Carine Tchout Himanshu sac	Total 20 20	Unreat ^ 19 19 1	5780 Monday, November 07, 202 ABAP at SAP Te Latent blog posts for ABAP THE premier toch confer THE premier toch confer inovators, and leaders solutions, platforms, and	Blog Posts Blog Posts Blog Posts Blog Posts Blog Posts Blog Posts Blog Post	1948 Contributo 2 Cronotouo Djono LP TechEd in 2022 is j Pers. IT practitioners, in gaining a detailed u	just around the c professionals, te understanding of	5 Followers	A States of
 type filter text Project / Feec √ (19) Na √ (19) Na √ (19) Na ↓ F L S	Following Write a Blog Post RSS tor feed entry Query / Feed Entry take Feeds Latest log posts for ABAP Development Bacomae a ald expert for PLAIFORM TECHNOLOGIES – Ask an Expert Peer Balk Read Feed entry: ABAP at SAP TechEd in 2022 mtr(07.11.2022 1425232, carine Choudoup Djorne) Arwangues courses storems write ymarry arranger uter CDS - Time Variance	Date 07.11.2022 07.11.2022 07.11.2022 07.11.2022 07.11.2022	Time 14:54:46 14:54:46 14:33:38 14:25:23 08:41:09 07:14:41	Author Ervin Szolke Carine Tchout Himanshu sac Jakob Mainka	Total 20 20	Unreat ^ 19 19 11 1	5780 Monday, November 07, 202 ABAP at SAP TE Latest biog posts for ABAP 1 Hey everyonel We are as THE premier tech confer innovators, and leaders solutions, platforms, and	Blog Posts Blog Posts Blog Posts Blog Posts Blog Posts Blog Posts Blog Post	1948 Contributo 2 Genetate Djeme NF TechEd in 2022 is is pers, If practice in 2022 is in gaining a detailed t	just around the c professionals, te understanding of	5 Followers	A S
type filter text Project / Feec (19) Na (19) Va E E S S	Following Write a Blog Post RSS tor feed entry I Query / Feed Entry Attive Feed Litest blog posts for ABAP Development Become a Paid expert for PLATFORM TECHNOLOGIES - Ask an Expert Peer Marka affeed entry: ABAP at SAP TechEd in 2022 thr Or11.2022 H24232 Carine TechEdd in 2022 Ary Unaryings Courses with S/HANK UPC DS - Time Variance pend Reporting with S/HANK Turbedded Analytics	Date 07.11.2022 07.11.2022 07.11.2022 07.11.2022 07.11.2022 07.11.2022 04.11.2022	Time 14:54:46 14:54:46 14:33:38 14:25:23 08:41:09 07:14:41	Author Ervin Szolke Carine Tchout Himanshu sac Jakob Mainka Uladzisłau Pra	Total 20 20	Unreat ^ 1 19 1 1 1	5780 Monday, Newember 07, 302 ABAP at SAP TE Hey everyone! We are a THE premier tach content reversion al leaders soutions, piatforms, and	Blog Posts 2 02-25 PM chEd in 20222 weekspmeric Carine T imost there: the SA tence for all develop there studies,	1948 Contributo 2 Crhodoso Djomo PP TechEd in 2022 is j pers, IT practitioners, in gaining a detailed u	just around the c professionals, te understanding of	5 Followers	A 8
type filter text Project / Feee ♥ (19) Na ♥ (19) Na	Following Write a Blog Post RSS tor feed entry IQuery / Feed Entry IQuery / Feed Entry Latest blog posts for ABAP Development Become a Paid expert for PLATFORM TECHNOLOGIES - Ask an Expert Peer BABA Bace feed entry ABAP at SAP Teched in 2022 Intro (711.1202 Lat2232, Carine Functionuo Dipmo AP Prangrues Courses wrom 3/emprove twoer DS - Time Variance ispend Reporting with S/HANAA Embedded Analytics How to create CDS View Report In RISE With SAP Public Cloud 2208	Date 07.11.2022 07.11.2022 07.11.2022 07.11.2022 07.11.2022 07.11.2022 07.11.2022 07.11.2022	Time 14:54:46 14:54:46 14:35:38 14:25:23 08:41:09 07:14:41 16:22:28 20:06:34	Author Ervin Szolke Carine Tchout Himanshu sac Jakob Mainka Uladzislau Pra Jakshmi naray	Total 20 20	Unreat ^ 19 19 11 11 11 11 11 11 11 11 11 11 11	5780 Menday, November 07, 2022 ABAP at SAP Te Latest tiop posts for ABAP THE premier tech confer innovators, and leaders southors, platforms, and	Blog Posts	1948 Contributo 2 Production Operno up TechEd in 2022 is P TechEd in 2022 is n gaining a detailed u	ust around the c	5 Followers	A 8

Figure 118 Subscribing to Popular RSS Feeds

With a left-click on the title in the detail display or right-click in the list and the context menu item Open you can open the blog article (for this it makes sense to set an external browser in Eclipse, because the Eclipse internal browser pretends to be IE11 for the website).

The system news feed displays the current messages from the system administrators.

You can add various filter options to feeds on runtime errors (ST22 dumps), including filtering by triggering user, responsible user, object and package user, or package. The filters can also be used with and/or

(all/any-) Links to a hierarchical filter tree can be arranged. You can open a runtime error in a separate editor view (context menu in the list, link in the details; the well-known long text format or an unformatted display is also available there). You can navigate directly to the source code lines of the triggering location and the call hierarchy directly in the ADT source code editor.

An example with multiple runtime errors in one feed:

						ŝ,	📓 🕶 👜 🖌	Ă	8
type filter text for feed entry						Monday, November 14, 2022 11:12 AM			
Project / Feed Query / Feed Entry	Date	Time	Author	Total	Unrea ^	Assignment error: Overwriting of a protected field.			^
v 🛃 (15) _en	21.11.2022	17:23:48		16	15				
🗸 🍇 (15) Runtime Errors	21.11.2022	17:23:47		16	15	Show in Runtime Error Viewer			
Assignment error: Overwriting of a protected field.	14.11.2022	11:12:39							
Assignment error: Overwriting of a protected field.	14.11.2022	11:08:18	and the second		1	Contents			
Assignment error: Overwriting of a protected field.	14.11.2022	08:58:36			1	Header Information			
Assignment error: Overwriting of a protected field.	14.11.2022	08:57:09			1	What happened?			
Assignment error: Overwriting of a protected field.	14.11.2022	08:56:44	and the second		1	Error analysis Information on where terminated			v
Assianment error: Overwriting of a protected field.	14.11.2022	08:53:25	and the second		1 ×	Source Code Extract			
<					>	<		>	

Figure 119 Multiple Runtime Errors within a Feed



Documentation on the SAP Gateway Error Log Feeds can be found in the PDF document from Note 1797736 - SAP Gateway Troubleshooting Guide and in the blog How to use the SAP Gateway Error Log in ADT.

You can add various filter options to feeds, including user, service, namespace or package. In order to cope with many entries, paging can be activated. In the detail display, you can jump directly to transaction /IWFND/GW_CLIENT for replay. In addition, you can navigate directly into the ADT source code editor analogous to the runtime errors, see the following example:

type filter text for feed entry						Frontend Error: Kein Service tur Namensraum ", Name
Project / Feed Query / Feed Entry	Date	Time	Author	Total	Unreal ^	SAP Gateway Error Log (all)
v 🔚 (4) en	21.11.2022	17:51:20		5	4	and another and (m)
4. Gateway error log entries for my user (21.11.2022	17:50:59		0	0	Content
4. Runtime Errors caused by me ()	21.11.2022	17:50:58		0	0	Header Information
Runtime Errors for objects I am responsible for (21.11.2022	17:50:59		0	0	Service Information
🗸 🔩 (4) SAP Gateway Error Log (all)	21.11.2022	17:51:20		5	4	Source Code Extract
Frontend Error: VokabAnnotatDatei 'UI_PREPAREFORBILLG_VAN', Vers. '0001',	21.11.2022	17:34:34			1	Active Calls/Events
Frontend Error: Kein Service für Namensraum ", Name 'UI_PREPAREFORBILLG', Vers	i 21.11.2022	17:34:33				Header Information
Frontend Error: Kein Service für Namensraum '', Name 'UI_PREPAREFORBILLG	21.11.2022	17:34:33			1	<u>neader information</u>
Frontend Error: Kein Service für Namensraum '', Name 'UI_PERFORMREPAIRS'	21.11.2022	17:34:31			1	Short Text Frontend Error: Kein Service für Namensraum ", Name
Frontend Error: Kein Service für Namensraum '', Name 'UI_PROCREPAIRQTAN	5 21.11.2022	17:34:29			1	Transaction ID AA7E06A764004420952DE059E6600002 (Dealey in Old Olicet)
System Messages	21.11.2022	17:50:58		0	0 ~	Traisaction ID AA/E00A/0199442C652DEC56P009B6B2 (Replay in Gw Client)

Figure 120 View of an SAP Gateway Error from the Error Log

In the context of enterprise events, you can add feeds to errors from event processing, for example, to see errors when passing the events to the event mesh. To do this, you can filter channel and user.

You can create feeds for the ATC results from tests in a central ATC check system, see Documentation (On-Premise/Cloud). Here, too, various filter options are available. In the feed list, you can navigate to the details and source code line of a result.

If you have BW/S4HANA systems, you can use the BW Job Repository Feed to check the status of various job types (e.g. DATAFLOWCOPY or DTP_LOAD) have it informed, cf. Documentary. You can then branch from the feed entries to view the job details.

Since the Feed Reader is the ideal tool for proactive monitoring of applications (e.g. during and after a go-live of a new application), it makes sense to consider the authorization for ADT not only for the development systems, but also to enable the developers to collect the feeds from test and production systems by means of authorizations.



4.6 **Documentation Links**

ABAP Debugger Concept:

https://help.sap.com/docs/BTP/5371047f1273405bb46725a417f95433/4ec365a66e3 91014adc9fffe4e204223.html

Troubleshooting Tools:

https://help.sap.com/docs/BTP/5371047f1273405bb46725a417f95433/4ecc7d3a6e3 91014adc9fffe4e204223.html

Syntax for Breakpoint Conditions:

https://help.sap.com/docs/BTP/5371047f1273405bb46725a417f95433/d878e676fe9 04eba9f4bb79193154092.html



5 Installation, Distribution und Update Strategies

ADT are based on the open source Eclipse framework, which is very well known in other programming languages. For installation and distribution, this has the advantage that the wheel does not have to be reinvented. Depending on the size of the company and the heterogeneity of a company's IT landscape, it is even possible that Eclipse-based development environments are already used and distributed in the company. In this case, it is advisable to continue using the existing infrastructure. This can also reduce any effort required for one or the other solution.

5.1 **Delimitations**

5.1.1 Installation Guide of SAP

SAP publishes its own installation guide (link). In addition to the manual installation method also listed here, it also lists other variants for locked-down environments with very limited Internet access. Since these are solutions for specific situations, they will not be discussed in the following sections.

5.1.1.1 Other tools with identical installation path

SAP also publishes other tools based on Eclipse, such as BW Tools (mandatory from BW/4HANA 1.0) and HANA Development Tools. However, since this guide focuses on ABAP development with ADT, we will not go into further detail about the other tools mentioned. However, most of the findings are transferable.

5.2 **Preparations**

For every ADT installation (sometimes also Eclipse without ADT) there are certain requirements. These are independent of the installation/distribution strategy and are described in detail in the above-mentioned ADT Installation Guide.

5.2.1 Java Development Kit and Java Runtime Environment

If only ABAP is developed with the installation (and in particular no Java, i.e. no plugin development to complement ADT, see Chapter 7 Plug-ins), no Java Development Kit (JDK) is necessary. A Java Runtime Environment (JRE) is sufficient. If the official



installer of Eclipse is used as a basis (see Chapter 6 Installation, Distribution and Update Strategies), this comes automatically.

It should be noted that Eclipse is now installed as a 64-bit application by default (often referred to as x64 or $x86_64$). Because the JRE must use the same architecture as the Eclipse installation, a 64-bit JRE is also installed.

5.2.2 Backend

In addition to the local ADT installation, in the case of development on an on-premise system, it must also be prepared for connection to ADT. These steps are also described in the official SAP ADT Configuration Guide . However, two essential steps are often overlooked, which is why it should be explicitly pointed out again here.

5.2.2.1 Web Services

For communication, ADT use special web services on the backend, which are delivered with the supported basic releases. The definition of the web services and the handler classes entered in them are one reason why the functional scope of ADT differs between the basic releases. These web services are inactive by default and must first be enabled. The current list can be found in the SAP Configuration Guide linked above.

At the time of this document, these are the following services:

- ABAP Docu (required)
 - $\circ \quad \textit{default_host} \rightarrow \textit{sap} \rightarrow \textit{public} \rightarrow \textit{bc} \rightarrow \textit{abap} \rightarrow \textit{docu}$
 - $\circ \quad \textit{default_host} \rightarrow \textit{sap} \rightarrow \textit{bc} \rightarrow \textit{abap} \rightarrow \textit{docu}$
- Error texts and element info (required)
 - $\circ \quad \textit{default_host} \rightarrow \textit{sap} \rightarrow \textit{public} \rightarrow \textit{bc} \rightarrow \textit{abap} \rightarrow \textit{toolsdocu}$
 - $\circ \quad \textit{default_host} \rightarrow \textit{sap} \rightarrow \textit{bc} \rightarrow \textit{abap} \rightarrow \textit{toolsdocu}$
- Sharing HTTP links (optional)
 - \circ default_host \rightarrow sap \rightarrow bc \rightarrow adt
- Web Dynpro (only necessary for WD developers)
 - $\circ \quad default_host \rightarrow sap \rightarrow bc \rightarrow webdynpro \rightarrow sap \rightarrow wdy_aie_vd_preview$

5.2.2.2 Permissions

In order to be allowed to use the aforementioned web services, the users must be authorized. In addition, RFC blocks and transaction codes are required.



SAP provides two roles as templates for this purpose:

- 1. \rightarrow developer role with all features \rightarrow Entwicklerrolle mit allen Features
- 2. → *View permissions for all features* → Anzeigeberechtigungen für alle Features

Details regarding the built-in authorizations and their purpose can be found in the SAP ADT Configuration Guide .

5.2.3 SAP GUI Installation

If an SAP GUI transaction start is to be possible on a back-end system (not available in SAP Public Cloud and SAP BTP ABAP Environment/Steampunk), a local SAP GUI installation is required. ADT do not supply them.

5.2.4 Visual Studio Redistributable

On Windows, the Visual Studio 2013 (VC++ 12.0) redistributable package is also required in exactly this version. Often, this dependency is already present due to other software already installed on the target device.

5.3 Technical Structure of an Eclipse Installation

As mentioned in previous chapters, an Eclipse installation consists of the following components:

- Installation folders (software-only packages and eclipse.exe)
- User-Settings or Configuration-Area (plug-ins, parts of the configuration)
- Workspace (user-specific part: settings, views, system connections, etc.)

The installation of Eclipse creates the installation folder and user settings. Using advanced techniques, parts of the workspace can already be pre-assigned.

Due to various problems, it is recommended to place all components in a folder that can be written to with normal user rights (i.e. **not** C:\Program Files\).

For example, a directory structure for the Eclipse installation might look like this:

C:\ADT\	Complete directory for ADT
C:\ADT\IDE	Directory for storage of the different Eclipse versions
ADT in Eclipse	

129

C:\ADT\IDE\2022-12	Unzipped files of Eclipse version 2022-12
C:\ADT\IDE\2023-03	Unzipped files of Eclipse version 2023-03
C:\ADT\WS	Directory for storing the various workspaces
C:\ADT\WS\2022-12	Directory for version 2022-12 workspaces

•••

This example is based on the assumption that the Eclipse installation is done by unpacking the zip files and that the versions should be used in parallel.

Since the workspaces with ascending versions are converted to the new version, the workspaces should be copied per version so that you can continue to use the older versions if necessary.

The extent to which workspaces are used sensibly is explained in more detail in chapter 3 "Working with Eclipse".

5.4 Plug-ins

Plug-ins such as ADT can be added to an existing Eclipse platform by specifying the update site in the Help \rightarrow Install New Software dialog. However, it is easier to use the Eclipse Marketplace if the plug-in

5.4.1 Eclipse Marketplace

The Eclipse Marketplace is hidden in the Help menu.



Help	
•	Welcome
?	Help Contents
2	Search
	Show Context Help
	Show Active Keybindings Ctrl+Shift+L
0	Tip of the Day
8	Tips and Tricks
D,	Collect Support Information
æ	Report Incident on SAP BTP, ABA Open the Eclipse Marketplace wizard
	Provide Feedback for ABAP Development Tools
	Cheat Sheets
۹	Eclipse User Storage >
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~	Perform Setup Tasks
R	Check for Updates
<b>6</b> -	Install New Software
\$	Eclipse Marketplace
۲	About Eclipse IDE
•	Donate
*	Contribute

Figure 121 Getting Started with the Eclipse Marketplace

Here you can search for plug-ins. As of 2022, the search term *ABAP* returned 11 hits in this example.



Eclipse Mark	cetplace			_		×
Eclipse Market	tplace				~	
Select solution: Press the "more	s to install. Press Install Now to proceed with installation. e info" link to learn more about a solution.					3
Search Recent	Popular Favorites Installed 💡 Giving IoT an Edge					
Find: PABAP	×	All Markets	~	All Categories	~	Go
	ADAY AUI EXTENSIONS 1.0.0	·				^
ABAP	Next ABAP extension for ADT Tools. Split from ABAP Favorites in order to keep giving the users a $\underline{more\ info}$	clean and to k	eep the core funct	tionality of Favorites a	lone,	
ADT Extensions	by <u>ABAPBlog.com</u> , Apache 2.0					
	ABAP abap eclipse adt					
★ 31	Installs: 2,62K (76 last month)				Install	_
	ABAP Favorites 1.0.60					
ABAP	A small plugin to create ABAP favorites list for transactions and urls. Editing of Dev Objects $\underline{more\ info}$	entries now av	vailable. It contain	s two views: Favorites	Favorite	s
FAVORITES	by <u>ABAPBlog.com</u> , Apache 2.0 <u>ABAP sap ABAP favorites favorites</u>					
★ 38	Installs: 3,91K (95 last month)				Install	
	ABAP Quick Fix 1.0.16					
	ABAP Quick Fix plugin for ADT Installation from Marketplace https://marketpla more info	ice.eclipse.org	/content/abap-qu	iick-fix Direct installat	ion	
QuickFIX	by <u>ABAPBlog.com</u> , Apache 2.0 <u>ABAP ABAP ADT abap eclipse sap</u>					
* 22	Installs: 1,88K (79 last month)				Install	
	11 matches. Browse for more solut	tions.				~
Marketpla	aces					
	<b>A</b>					
						_
Ø		< Back	Install Now >	Finish	Cancel	

Figure 122 Sample Search for Plug-ins in the Eclipse Marketplace

Each plug-in has its own install button on the right-hand side. If necessary, a license must be confirmed and certificates trusted. Finally, a restart of Eclipse is necessary.

### 5.4.2 Update Site

If you know the update site of a plug-in or if it is not listed on the Eclipse Marketplace, the classic installation method can also be used.



Help	3	
• 🚳	Welcome	
?	Help Contents	
22	Search	
	Show Context Help	
	Show Active Keybindings	Ctrl+Shift+L
0	Tip of the Day	
8	Tips and Tricks	
D,	Collect Support Information	
<b>a</b>	Report Incident on SAP BTP, ABAP Environment	
	Provide Feedback for ABAP Development Tools	
	Cheat Sheets	
۹	Eclipse User Storage	>
~??	Perform Setup Tasks	
<i>e</i>	Check for Updates	
<b>6</b> 3.	Install New Software	
٩	Eclipse Marketplace	
۲	About Eclipse IDE	
٠	Donate	
*	Contribute	

Figure 123 Installation of New Software via the Context Menu

The update site is entered in the *Work with* field. If this is valid, the plug-ins available there are displayed below. In addition to web pages, a downloaded version of the plug-in in a zip file can also be an update site. In the latter case, however, updates must be carried out manually with another download.



Install			
Available Software			
Check the items that you wish to install.			
Work with: https://tools.hana.ondemand.com/latest		<ul> <li><u>A</u>dd</li> </ul>	<u>M</u> anage
type filter text			Select All
Name	Version		Deselect All
>      IDU ABAP Development Tools			
Details			
Show only the latest versions of available software	✓ <u>H</u> ide items that are already installed		
Group items by category	What is already installed?		
Show only software applicable to target environment			
Contact all update sites during install to find required software			
(?)	< <u>B</u> ack <u>N</u> ext >	<u>F</u> inish	Cancel

Figure 124 Entering the Update Site

Here, too, licenses may have to be accepted and certificates trusted. After a restart, the plug-in is available.

# 5.5 Installation and Distribution Strategies

### 5.5.1 Overview and Comparison

Since ADT were added to the Eclipse ecosystem comparatively late, various installation strategies for Eclipse-based development environments have already been established outside the SAP environment. The most common are:

- 1. Completely manual setup
- 2. Pre-configured initial installation
- 3. Eclipse Installer (Oomph)



In the following chapters, these mechanisms are discussed in detail. In addition to these, there are many other options that were either designed for rather unusual situations (e.g. lack of Internet access) or have now been replaced by convenient variants.

 $\circ$  + (good),  $\circ$  (medium) and - (bad) are to be understood as a ranking and not as absolute values at the beginning, middle and end of the respective spectrum. The reason for this is that, for example, an effort is perceived very individually. Thus, a high effort marked with - (badly) can still be acceptable for a person. However, these symbols represent a sequence since measurable differences exist.

Criterion	Manual Installation	Pre-configured Initial Installation	Eclipse Installer
User Effort	-	ο	+
Necessary Knowledge User	-	ο	+
Effort Central Administration	+ (none)	ο	-
Required Central Infrastructure	+ (none)	ο	-
User Effort for Update/Upgrade	0	ο	ο
Effort Central Administration for Update/Upgrade	+ (none)	-	ο
Automatic distribution Add- ons, Settings	- (impossible)	o (once)	+ (continuously)



Criterion	Manual Installation	Pre-configured Initial Installation	Eclipse Installer	
Recommended fo	r Individuals/small	Medium-sized and	Large companies	
Company Size	businesses	large companies		

 Table 1 Comparison of Different Installation Options

### 5.5.2 Manual Installation

In this variant, a standard Eclipse installation is re-downloaded and installed with just a few clicks. Since there is no ready-made installer for ABAP, ADT are installed individually. Without plug-ins, settings and perspective adjustments, a usable installation is available after about 15 minutes.

SAP has provided illustrated instructions on its learning platform: https://developers.sap.com/tutorials/abap-install-adt.html.

Once this installation is complete, the desired settings must be made and the system connections must be added. If further plug-ins are used (see chapter 7 "Plug-ins"), they must also be installed.

### 5.5.3 **Pre-configured Initial Installation**

A preconfigured initial installation is basically a manual installation, the state of which is backed up immediately after setup (usually packaged as a zip file). This condition can then be distributed to other people in various ways.

With macOS as the target system, it is also important to note that a so-called app-zip translocation could take effect here. This creates a kind of "shadow copy" when executed by software such as Eclipse, which has just been unpacked *in the same directory*. The result behaves like missing write permissions in the installation folder. If updates are installed, errors occur.

Depending on the intended use, more or less of the three components of the installation (installation folder, user settings, workspace) can be packed in.

If you take the user settings with you, ADT and its update site, for example, can be distributed at the same time for later updates. In Citrix environments, on the other hand, this would be a hindrance, as the user settings require write permissions. If you take two separate packages for this, the installation folder could also be placed in a read-only, centrally provisioned part.



The workspace could also be packaged and distributed as a separate template. This means that initial settings can be distributed once. However, since this area changes very frequently and cannot be used without losing the current settings, layouts and open objects during an update anyway, this should only be done with caution.

### 5.5.4 Eclipse Installer

On the Eclipse website, a convenient installer with a very small download size can be downloaded (also called *Oomph Installer*). When you start it, you can then select a basic configuration and a few detailed settings and then have exactly this installation created.

However, there are no configurations including ADT here. However, this is not magic, but the available settings are only stored in a certain format on a server of the Eclipse Foundation. The path where the installer should search for configurations is customizable. Thus, it is possible to create company-specific configurations that anyone can install at will. This method could already be used in some companies for other development languages, which significantly reduces the maintenance effort.

Only the Eclipse installer has to be distributed, for example, via a software distribution mechanism together with the setting where the configurations can be found. In addition, this variant offers the option of presetting individual settings in the workspace and keeping them up-to-date.

The disadvantage of this variant is the comparatively high central effort. It is therefore not suitable for single/few standard installations.

The Oomph project has provided an extensive documentation in English, which is well suited as a reference work: Link.

### Administrator Information

The following sections explain how to create and adjust the Oomph configurations. This is usually done by a few administrators.

### Definitions

If you read through Oomph's documentary, you will first be overwhelmed with many new terms. Therefore, here are the most important terms used in the following sections.



Term	Description
Setup Model	The Oomph configurations, as in most programming languages, are file-based. A set of these files with a specific format is called a setup model.
Product	An installation-level configuration (platform with specific version + plug-ins)
Project	Project-specific settings. In the git-based world, for example, this can be the default of standard repositories. Workspace-level configuration.
Index	The library of available configurations that can be selected in the Eclipse Installer.

#### Table 2 Terms in Oomph

An installation using Eclipse/Oomph Installer always installs a platform, the plug-ins, Oomph Updater, Oomph Recorder and finally the project settings in the workspace.

### Required software for administrators

If you want to design and manage your own configurations, you need the Oomph SDK. This is nothing more than a set of plug-ins on a standard Eclipse platform. This delivers the required views and ready-made perspectives in order to be able to edit the configuration files graphically (tree views, forms, etc.).

### Step by step to a basic configuration

When creating configurations, one usually proceeds from general to specific. For all the required files mentioned below, the Oomph SDK offers Wizards (File  $\rightarrow$  New  $\rightarrow$  Other  $\rightarrow$  Oomph  $\rightarrow$  ...).

### Index

This means that you first create an *index*. An index refers to available *product catalogs* and *project catalogs*. By default, the index is named *org.eclipse.setup*, but it can also *be called* myFirst.setup, for example.



The following is the structure of an index:
xml version="1.0" encoding="UTF-8"?
<setup:index< td=""></setup:index<>
xmi:version="2.0"
xmlns:xmi="http://www.omg.org/XMI"
xmlns:setup="http://www.eclipse.org/oomph/setup/1.0"
name="myCompany Eclipse Setups"
label="index">
<productcatalog< td=""></productcatalog<>
href="myCompany.products.setup#/"/>
<projectcatalog< td=""></projectcatalog<>
href="myCompany.projects.setup#/"/>

In the extended view of the Eclipse Installer you also have the possibility to switch between several available indexes.



S Eclipse Installer	– <b>D</b> X
Product	
Select the product and choose the version you want to install.	Oomph
	😹 🕂 🗙 🖻 🗀 👻
Comparison	<user products=""></user>
IDE for Java Developers	Eclipse Products
	Switch Catalog Index > • Eclipse Setups
	Developer Setups
	Manage
Eclipse Products	
Eclipse Produkte für	
Product Version:	×
Java VM:	
Bundle Pool: C:\ProgramData\eclipse\.p2\pool	<u> </u>
⑦ ▲ ● ⑨ € ◎          ● 1.25.0 Build 5517         < Back         Next >         Emili	sh Cancel

#### Figure 125 Switching between Indices

### Product Catalog and Product

A product catalog lists various products. It also contains overarching settings, such as:

- internal redirects of update sites to local caches
- Definition of variables that can later be filled differently for each product (version) or project (for example, for version-specific update sites)
- the installation of the Oomph Client for later distribution of updates to the settings specified in the Products and Projects, Oomph Recorder (see section <u>User Information</u>).

The following properties can be stored at all levels (Product Catalog, Product and Product Version):



# Installation, Distribution und Update Strategies

New Child Image: Product   Undo Ctrl+Z   Redo Ctrl+Y   Cut Image: Paste   Copy Image: Paste   Delete Image: Paste   Delete Image: Paste   Find/Replace Image: Paste   Image: Version Paste Image: Paste   Image: Paste Image: Paste   Delete Image: Paste   Image: Paste Image: Paste <t< th=""><th></th><th></th><th></th><th></th><th></th></t<>					
Undo       Ctrl+Z       Compound         Redo       Ctrl+Y       K       Variable         Cut       ->       Redirection         Copy       Stallation       Stallation         Paste       Stallation       Vorkspace         Delete       Stallation       Stallation         K       Delete       Stallation         K       Find/Replace       Stallation         K       Find/Replace       Stallation         K       Find/Replace       Stallation         K       Find/Replace       Stallation         K       Coverage As       Stallation         K       Control       Stallation         K       Projects Build       Projects Build         K       Stallation       Stallation         K       Stallation       Stallation         K       Stallation	\$	New Child	>	۲	Product
Redo       Ctrl+Y       64       Variable         Cut        Redirection         Copy       Sig       Installation         Paste       Sig       Workspace         Delete       Sig       Ctrl-Y         Find/Replace       Sig       Ctrl-Y         Vive Validation       Sig       Cit Clone         Show Tooltips       Sig       Cotrol         Control       Sig       Paterece         Control       Sig       Projects Build         Open in Setup Editor       Sig       Projects Build         Open in Text Editor       Sig       Projects Build         Show Unformation Browser       Sig       String Substitution         Show Working Sets Preview       String Substitution         Show Properties View       Fing Substitution	Ý	Undo Ct	trl+Z	C	Compound
Cut       Image: Cut	\$	Redo Ct	trl+Y	(×ĵ²	Variable
Copy       Mathematical installation         Paste       Mathematical installation         Delete       Mathematical installation         Find/Replace       Elipse Ini         Live Validation       Git Clone         Show Tooltips       Mathematical installation         Coverage As       Mathematical installation         Control       Mathematical installation         Open in Text Editor       Mathematical installation         Projects Build       Projects Import         Projects Import       Resource Creation         Projects Suild       Projects Installation         Projects Suild       Projects Installation         Projects Suildinon       Projects Installation <th>ď</th> <th>Cut</th> <th></th> <th>÷</th> <th>Redirection</th>	ď	Cut		÷	Redirection
Paste       Morkspace         Delete       Image: Cond         Find/Replace       Image: Cond         Image: Control       Image: Control         Image: Control       Image: Cond         Image: Cond       Image: Cond         Image: Cond       Image: Cond         Image: Cond		Сору		81	Installation
DeleteImage: CondFind/ReplaceImage: Chipse IniFind/ReplaceImage: Chipse I	Ē	Paste		îê	Workspace
Find/Replace       Image: Eclipse Ini         Live Validation       Git Clone         Show Tooltips       Image: Link Location         Show Tooltips       Macro Expansion         Coverage As       Image: P2 Director         Control       P3         Load Resource       Image: Projects Build         Open in Setup Editor       Projects Build         Open in Text Editor       Projects Import         Refresh       Resource Creation         Show Working Sets Preview       Image: Preview         Show Properties View       Image: Preview         Show Properties View       Additional Tasks	ж	Delete		2	Cmd
Live Validation       Image: Constant of the constant	ß	Find/Replace			Eclipse Ini
<ul> <li>Live valuation</li> <li>Show Tooltips</li> <li>Coverage As</li> <li>Control</li> <li>Control</li> <li>Load Resource</li> <li>Open in Setup Editor</li> <li>Open in Text Editor</li> <li>Refresh</li> <li>Show Information Browser</li> <li>Show Working Sets Preview</li> <li>Show Properties View</li> <li>Show Properties View</li> <li>Autor Expansion</li> <li>Macro Expansion</li> <li>Macro Expansion</li> <li>Macro Expansion</li> <li>Macro Expansion</li> <li>Path Variable</li> <li>Preference</li> <li>Projects Build</li> <li>Projects Import</li> <li>Resource Copy</li> <li>Resource Creation</li> <li>String Substitution</li> <li>String Substitution</li> <li>Show Properties View</li> <li>Macro Expansion</li> <li>Additional Tasks</li> </ul>	2	Live Validation		GIT	Git Clone
Show roottips       Macro Expansion         Coverage As       P2 Director         Control       Path Variable         Load Resource       Preference         Open in Setup Editor       Projects Build         Open in Text Editor       Projects Import         Refresh       Resource Copy         Show Information Browser       String Substitution         Show Working Sets Preview       Text Modify         Show Properties View       Additional Tasks		Cheve Talakina		Ê	Link Location
Coverage As       Image: Coverage As       Image: P2 Director         Control       Image: Path Variable       Image: Preference         Load Resource       Image: Projects Build       Image: Projects Build         Open in Setup Editor       Image: Projects Import       Projects Import         Open in Text Editor       Image: Projects Import       Projects Import         Show Information Browser       Image: Project Preview       Image: Preview         Show Vorking Sets Preview       Image: Preview       Image: Preview         Show Properties View       Image: Preview       Image: Preview	P	Show looitips		Ŷ	Macro Expansion
Control       Image: Control       Path Variable         Load Resource       Preference         Open in Setup Editor       Projects Build         Open in Text Editor       Projects Import         Refresh       Resource Copy         Show Information Browser       Prige Substitution         Show Vorking Sets Preview       Pext Modify         Show Properties View       Additional Tasks	Q_	Coverage As	>	<b>6</b> 3	P2 Director
Load Resource       Preference         Open in Setup Editor       Projects Build         Open in Text Editor       Projects Import         Open in Text Editor       Resource Copy         Refresh       Image: Show Information Browser       Image: Show Vorking Sets Preview         Show Vorking Sets Preview       Image: Text Modify         Show Properties View       Additional Tasks	9	Control		Ċ	Path Variable
Open in Setup Editor       Projects Build         Open in Text Editor       Projects Import         Refresh       Resource Copy         Show Information Browser       String Substitution         Show Vorking Sets Preview       Projects Build         Show Properties View       Additional Tasks	<u>8</u> .	Load Resource		8-	Preference
Open in Setup Editor       Projects Import         Open in Text Editor       Resource Copy         Refresh       Image: Copy Copy Copy         Show Information Browser       Image: Copy Copy Copy         Show Working Sets Preview       Image: Copy Copy Copy         Show Properties View       Image: Copy Copy Copy         Show Properties View       Image: Copy Copy Copy         Show Properties View       Image: Copy Copy Copy         Image: Copy Copy Copy Copy Copy Copy Copy Copy	_	On an in Satur Editor		<b>™</b>	Projects Build
Open in lext Editor     Resource Copy       Refresh     Image: Resource Creation       Show Information Browser     Image: String Substitution       Show Working Sets Preview     Image: Text Modify       Show Properties View     Additional Tasks	// 	Open in Setup Editor		Ê	Projects Import
Refresh     Resource Creation       Show Information Browser     String Substitution       Show Working Sets Preview     Text Modify       Show Properties View     Additional Tasks	Z	Open in Text Editor		12	Resource Copy
<ul> <li>Show Information Browser</li> <li>Show Working Sets Preview</li> <li>Show Properties View</li> <li>Additional Tasks</li> </ul>	<b>%</b>	Refresh		ľ	Resource Creation
<ul> <li>Show Working Sets Preview</li> <li>Show Properties View</li> <li>Additional Tasks</li> </ul>		Show Information Browser		₿¢₽	String Substitution
Show Properties View Additional Tasks	E	Show Working Sets Preview		D	Text Modify
		Show Properties View			Additional Tasks

Figure 126 Adding and Setting Properties

The most needed elements are:

Product	Adds a product to the product catalog (for example, IDE for JAVA Developers). Can only be added at the Product Catalog level.
Product Version	Adds a version to a product in which version-specific settings can then be made. Can only be added at the Product level.
Compound	This is a kind of folder in which settings can be grouped later.
Eclipse ini	Adds an option to the Eclipse.ini.
Variable	Variable with value assignment. It can be referenced later.



Redirection	URL redirect. Thus, the installations can integrate the original sources, but in fact, for example, an Artifactory is accessed.
P2 Director	List of Feature Groups to install.
Repository	Provides an installation source for Feature Groups

#### Table 3 Key Elements

For a minimal ADT installation, the following is required::

- One Product, z. B. "SAP"
- One Product Version, z. B. "2022-03 (4.23)"
- One P2 Director Task with:
  - Eclipse Platform Packages
    - epp.package.java (Value-Range starts at desired release  $\rightarrow$  4.23)
    - org.eclipse.platform (Value-Range starts at desired release  $\rightarrow$  4.23)
    - org.eclipse.rcp (Value-Range starts at desired release → 4.23)
    - org.eclipse.buildship
    - org.eclipse.tips.feature
    - org.eclipse.epp.mpc
- Repository-URLs for Eclipse Platform Packages
  - https://download.eclipse.org/releases/2202-03/202203161000 (Link release-dependent!)
  - https://download.eclipse.org/technology/epp/packages/2022-03/202203101200 (Link release-dependent!)
  - The most recent version is available via https://download.eclipse.org/technology/epp/packages/latest/

- > 🜆 BrandingInfo
- - 🚸 epp.package.java [4.23.0,5.0.0) ([(&(osgi.arch=aarch64)(osgi.os=linux)(osgi.ws=gtk))(&(osgi.arch=aarch64)(osgi.os=macosx)(osgi.ws=cocoa))(&(osgi.arch=
  - 🖗 org.eclipse.rcp [4.23.0,5.0.0)
  - 🖗 org.eclipse.buildship
  - org.eclipse.tips.feature
  - org.eclipse.epp.mpc
  - > 📢 https://download.eclipse.org/releases/2022-03/202203161000
    - https://download.eclipse.org/technology/epp/packages/2022-03/202203101200

#### Figure 127 Components of a "minimal" ADT Installation



^{🗸 🌒} SAP

### Project Catalog and Project

The Project Catalog lists available projects. The latter contain instructions for changing the workspace. For example, the plug-ins to be installed are also stored here.

The most important elements in a project are:

P2 Director	Grouped requirements and repositories
Requirement	Feature Group to install
Repository	Update Site URL
Stream	Obligatory object. Different configurations per stream possible. Can be present with a name, but without content.
Variable	Variable string with referencing, e.g. in repositories
Eclipse ini	Changes to the Eclipse.ini
Preference	Preset change of settings (Window $\rightarrow$ Preferences).

### Table 4 Most Important Elements of a Project

For a minimal ADT installation, the following is required:

- P2 Director Nodes
  - ADT Feature Groups
    - com.sap.adt.tools.hana.devedition
    - com.sap.core.devedition
  - Repository for ADT
    - https://tools.hana .ondemand.com/latest
    - Alternativ:  ${Variable} \rightarrow z. B. {sap.repository.url}$
- An empty stream, "Master" by default





#### Figure 128 Components of a Minimal ADT Installation

It should be noted that a plug-in for Eclipse can consist of several feature groups. How to find out is explained in the section Installing an additional plug-in .

#### Distribution of the installer and configuration

The Eclipse installer is the same exe file that can be downloaded from the Eclipse home page. This must be distributed to the end devices.

The configuration is distributed by creating/customizing a file with a specified name and path:

C:\Users\<currentUser>\.eclipse\org.eclipse.oomph.setup\setups\indices.xmi

Example of an index list:

```
<?xml version="1.0" encoding="UTF-8" ?>
```

<base:Annotation

```
xmi:version="2.0"
```

xmlns:xmi="http://www.omg.org/XMI"

xmlns:base="http://www.eclipse.org/oomph/base/1.0"

source="IndexLocations">

<detail key="https://pages.github.com/path/subpath/myFirst.setup">

<value>Description of my first Oomph Catalog</value>

</detail>

</base:Annotation>


The path to one or more indexes must be specified. The path can also be a web link or git repository path. The linked files will then be downloaded/updated each time the Eclipse installer is started.

Now any user in possession of the installer can select and install these configurations. Alternatively, the installer can also be distributed within the company via software distribution. Thus, a synchronization of the versions between Oomph components in the installations and the Eclipse installer used could be achieved.

#### Adaption Examples

So far, minimal configurations for installing Eclipse and SAP ADT have been described. However, the strengths of using the Oomph Installer only become apparent in the further pre-configuration during installation. The most common expansion requests will now be addressed.

### Installing an additional plug-in

One of the advantages of Eclipse as a development platform is its openness to extensions. Thus, plug-ins can be written by third-party manufacturers or dedicated community members that extend the functionality of ADT even further.

These can be automatically pre-installed in all installations in an Oomph Project. Unfortunately, however, it is not possible to add a repository as an update site, but then not install a plug-in on it. These update sites will be discarded when Oomph is installed (as of 2022).

Now, a plug-in consists of one or more feature groups. The latter must be deposited in the Oomph Project. However, the name is usually not known. For this purpose, there is a view "Repository Explorer" in the Oomph SDK (Window  $\rightarrow$  Show View  $\rightarrow$  Other  $\rightarrow$  Oomph  $\rightarrow$  Repository Explorer). By (manually) specifying an update site, a query of the provided feature groups is performed. These can then be done by drag-anddrop or **CTRL+C** and **CTRL+v** in the P2 Director node of the Oomph Project. In addition to the feature groups, there must also be a repository node that makes this update site available (can also be done more generally, i.e. in the Project Catalog or Product (-Catalog)).

As of October 2022, the query of the SAP update site looks like this:



htt	ps://tools.hana.ondemand.com/latest
ty	pe filter text
~	100 ABAP Development Tools
	ABAP Business Object Tools (Developer Edition)
	🖗 ABAP Core Development Tools (Developer Edition)
	ABAP Development Tools for SAP HANA (Developer Edition)
	🖗 ABAP Development Tools for Web Dynpro (Developer Edition)
	Real ABAP Enterprise Services and Integration Development Tools (Developer Edition)
>	100 Modeling Tools for SAP BW/4HANA and SAP BW powered by SAP HANA
>	IOI SAP Cloud Business Application Tools
5	IOI SAP HANA Tools

Figure 129 Components of the SAP Update Site

#### Specifying settings

In addition to the installation of ADT, one of the biggest advantages is the presetting of various settings for all installations.

All settings to be assigned are placed in the Compound/*User-Preferences* folder of the Oomph Project. Below that, there are individual subfolders per feature (as part of a feature group) and then the settings. However, it is up to each feature to decide in which representation it stores its settings. Some settings are stored as direct values (classic checkboxes), others, for example, as one large XML per settings page. Especially with the latter, only all settings of the page or none can be specified. An example of XML representation is the code templates of SAP ADT.

Since this non-uniform display is impractical to administer and the names of the features are usually not known, there are also auxiliary tools here: An oomph recorder is installed in the settings. For more information on how to use it as a user, see the Oomph Recorder section. The Oomph Recorder records the latest status of all changed settings at the user level (i.e. across Eclipse installations) and asks if they should be saved always/once/never after closing the settings. It also makes a button available in the button bar of Eclipse to view the settings that have already been saved.



 ws - InstallerSetup/COMPH/setups/

 File
 Edit
 Source
 Refare
 Navig

 Image: Source
 Refare
 Navig

 Image: Source
 Refare
 Navig

 Image: Source
 Refare
 Navig

 Image: Source
 Refare
 Navig

Figure 130 Ability to View the Settings that have Already Been Saved

✓ Å User
InstallationTask.location = \${install.root/}\${installation.id}
₩ WorkspaceTask.location = \${installation.location/ws}
✓
✓ Im UserPreferences
📖 /instance/com.sap.adt.debugger/systemDebugging -> ignore
📖 /instance/com.abapci.plugin/coloredProjectsStatusBarEnabled -> record
📼 /instance/com.abapci.plugin/coloredProjectsTitleIconEnabled -> record
📖 /instance/com.abapci.plugin/coloredProjectsStatusBarWidgetEnabled -> record
🗸 🧰 com.abapci.plugin
coloredProjectsStatusBarEnabled = true
coloredProjectsStatusBarWidgetEnabled = false
ColoredProjectsTitleIconEnabled = false
(x)= install.root = C:\Software\eclipse (default: C:\ProgramData\eclipse)

#### Figure 131 Settings that Have Already Been Saved

Again, these are setup files. This means that the recorded settings can be **transferred to the Oomph Project** with drag-and-drop or CTRL+C and **CTRL+V**.

The following are a few examples of generally distributed settings. Here are some of ADT-exclusive features, all of which SAP unfortunately does not enable by default.

- Code Templates
- Activation of additional code highlights in ABAP (different colors)
- Enable continuous syntax checking
- Activating the Occurrence Marker
- Uniform display of indents (tabs or spaces, width of indentation)

#### Increasing memory limits

With SAP ADT, there are no longer any limits to the number of open windows/objects. However, this also means that with many simultaneous tabs, the required RAM



skyrockets. Since Eclipse itself is also a Java application, VM parameters must be set. If the available memory of a Java VM is exceeded, a swapping in hard disk files takes place analogous to the RAM management of the operating system. This makes Eclipse slow and the fun of using it suffers. Minimum and maximum limits, initial reserved memory and much more can be set for VM memory. However, since this can also break a lot and the required limits are very individual, concrete recommendations are deliberately omitted.

If such parameters are specified for all users, they must be specified in the Eclipse .ini. Thus, corresponding Eclipse Ini tasks must be created in the Oomph Project.

# First start with ABAP Perspective

If the SAP ADT is installed, the standard perspectives are initially only available, but not opened. Eclipse, for example, starts with the Java Perspective, which most ABAP developers will rarely need. Therefore, it is desirable to start the installation directly with the delivered ABAP Perspective. Unfortunately, it is not possible to store additional perspectives (e.g. debug, ABAP profiling, etc.) in the most recently used perspectives in the upper right corner.

Forcing a start perspective is a start option in the Eclipse .ini. Therefore, an Eclipse Ini Task with the following properties must be created:

Option	-perspective
Value	com.sap.adt.ui.AbapPerspectiv e
VM	false

Table 5 Features of Eclipse Ini Task

5.5.4.1 User Informationen

Installation via Eclipse Installer

When the Eclipse installer is started for the first time, it may start in simple mode. However, the extended mode is required to use your own configurations. You can switch to it via the menu at the top right ("Advanced Mode").



eclipseinstaller by Domph	×	
type filter text	Q	♥ UPDATE         ADVANCED MODE         BUNDLE POOLS         WEB LINKS         TRUST         MARKETPLACE         ASK A QUESTION         REPORT A PROBLEM         ★ CONTRIBUTE         ABOUT         EXIT

Figure 132 Switch to "Advanced Mode"

Now you can see a list of available products. This is the content of the first referenced index in the Indices.xmi. In the upper right corner you can switch between all listed indices.



🚯 Eclipse Installer  $\times$ ٩ Product Select the product and choose the version you want to install. Oomph 🗞 🕂 🗙 🖻 🗁 🗸 Career Eclipse Products
 Total International Internatione International Internatione International International Interna <User Products> Eclipse Products Switch Catalog Index Eclipse Setups > • . CI/CD Catalog Developer Setups Manage... Eclipse Products Eclipse Produkte für Product Version: Java VM: ~ = Bundle Pool: C:\ProgramData\eclipse\.p2\pool ~ ? 🌢 🥯 🛞 🗄 🎯 🚳 <u>1.25.0 Build 5517</u> < Back Next > Finish Cancel

### Installation, Distribution und Update Strategies

Figure 133 Switching between Indices

Here you select a suitable entry. Once this is done, the available product version will be selectable at the bottom.



Eclipse Installe	D Eclipse Installer			
Product Select the produc	ct and choose the version you want to install.		Comph	
type filter text		-	× 🗆 🗀 🗸	
SAP	ducts> se Products			
My Product prov	rides cool stuff.			
Product Version:	2202-06 (4.24)		~	
Java 11+ VM:	C:\Program Files\Java\OpenJDK11 (Current)		~ =	
Bundle Pool:	C:\ProgramData\eclipse\.p2\pool		~ 🖄	
? 🌢 🤛	(₹) (€) (€) (€) (€) (€) (€) (€) (€) (€) (€	Finish	Cancel	

Figure 134 Selection of the Product Version

In the next picture, the projects can be selected. Theoretically, several projects can be selected for one installation. However, it can happen that these projects then make competing settings, and problems arise.

A stream must then also be selected for a project. If only one has been defined, it is already preselected.



S Eclipse Installer				_	
<b>rojects</b> Check the projects you want to provision, and for each choose its stream in the table column.					Oomph
type filter text				+	× 🗆 🗀 •
	Û	Ŷ			
Catalog	Project		Stream		
Projects	🗁 <user> - ABAP</user>		😞 Master		
? 🌢 🧼 🔋 🗄 🌚 🐁 1.2	5.0 Build 5517	< Back	Next >	Finish	Cancel

Figure 135 Selection of the Stream

Now defined and used, but not filled variables are queried. These can be, for example, the paths to the installation and workspace.



🚯 Eclipse Installer					
Variables					
Enter values for the required	d variables.				Oomph
Installation location rule:	\${install.root/}\${installation.id}				~
Installation folder name:	ABAP-Master				
Root install folder:	C:\Software\eclipse				Browse
Workspace location rule:	\${installation.location/ws}				~
Show all variables					
? 🔺 🤛 疧 🗐	) 🛞 🛞 <u>1.25.0 Build 5517</u>	< Back	Next >	Finish	Cancel

#### Figure 136 Querying other Variables

Finally, a confirmation page is displayed, but it usually does not contain any new information.

During installation, pop-ups may appear asking which license terms to accept and/or trust certificates.

If the check mark is set in the installer, the instance that has just been installed will be started after the installation. The settings of the workspace are then preconfigured again, since the workspace only exists now. Now the installation is complete.



# Oomph Recorder

The Oomph Recorder is a tool to unify and save settings across Eclipse installations. Optionally, these settings can also be loaded to the Eclipse User Account, so that cross-device synchronization is also possible here. In most companies, however, the use of an Eclipse account is probably not possible or not welcome.

The Oomph Recorder is activated under Window  $\rightarrow$  Preferences with a new recording icon. From then on, every time you close the settings, you will be asked whether the settings that have just been changed should always/once/never be saved in the last state.

These user-level settings are applied to a new Eclipse installation only after the first start and after the settings of the Oomph Project used. This means that standard settings can also be overridden on a user-specific basis or your own default settings can be defined.

# 5.6 Error Situations

During installation or an update, various errors may occur. The most common mistakes will be discussed here.

# 5.6.1 Errors during update or upgrade

When performing an update or upgrade, the installation may fail.

An installation usually takes place in the following phases:

- 1. Calculate target versions of all components and their packages, including consideration of dependencies
- 2. Download Packages
- 3. Installing Packages

Experience has shown that the most common cause of error in the calculation of target versions is dependencies that cannot be fulfilled.

For example, in a made-up example, an ADT upgrade also requires a newer version of Eclipse itself. However, only the current version can be found in the available update sites. Thus, the cause of the error here is an incorrect or outdated update site.



In more difficult cases, for example, in newer releases of Eclipse, a component may be removed or replaced. However, a plug-in like ADT still has a dependency on it. Only the manufacturer of the plug-in can remedy this.

Such a situation can arise when using HANA Studio as the Eclipse platform. Here, no public update sites for the Eclipse platform are set up during installation, which is why it no longer fits with current ADT versions after six months (= two quarterly releases) at the latest.

Errors occur much less frequently during the download. In the first step, the inventory lists of the update sites (artifacts.xml and contents.xml) were examined and a suitable package version was found. However, if this is not in the corresponding subfolder, a download error occurs. In this case, the update site is inconsistent. This can also only be fixed by the provider of the package.

# 5.6.2 Single-sign-on Libraries

Depending on the single sign-on strategy, environment variables on libraries may be necessary. If SAP GUI is only installed as a 32-bit application (up to SAP GUI 7.70 the only option), the libraries referenced in the environment variables SNC_LIB and SNC_LIB_2 are always used.

However, if Eclipse is installed as a 64-bit application (the default case), the SSO library referenced in the SNC_LIB_64 environment variable is used.

In older blog entries, the Kerberos libraries of Windows are still recommended (32 bit: gsskrb5.dll, 64 bit: gx64krb5.dll). However, these are no longer present in newer Windows installations and should no longer be used. SAP delivers an SAP CryptoLib with the SAP GUI. This is available in 32 bits and 64 bits.

#### 5.6.3 "No repository found containing"

https://launchpad.support.sap.com/#/notes/2186770

Every now and then there seem to be problems updating ADT. Several errors appear in the log: "No repository found containing: ...". The note recommends removing the update site, restarting Eclipse, and then re-adding the update site.

# 5.6.4 PKIX - Certificate Error

https://launchpad.support.sap.com/#/notes/3131747



This is a certificate error. An encrypted connection (HTTPS) is established to the update site. If SSL connections are broken in the corporate network or there is no common keystore for internal and external update sites, then this error can occur. The note gives a possible solution to this. Another option is to distribute your own custom JDK.

# 5.6.5 macOS aarch64 support & SAP GUI for Java

https://launchpad.support.sap.com/#/notes/3251738

The architecture of the SAP GUI and Eclipse installation should always be installed with the same processor architecture. Especially with Apple M1 and subsequently, there could be deviations here.

# 5.6.6 Offline-Installation – Download ADT-Dependencies

https://launchpad.support.sap.com/#/notes/2369308

If ADT is to be installed offline, various dependencies must be taken into account. This note gives some remedies in this case.



# 6 <u>Best Practices Eclipse Configuration</u>

# 6.1 Settings in Eclipse

In Eclipse, there are numerous settings that can make life as a developer easier, but sometimes harder. In this section, you will learn more about the different options and the most important settings. Important to know: There are two levels at which you can make settings. The *global level for* Eclipse and the *project-specific level* for an SAP system.

### 6.1.1 Globale Settings

After opening the settings via the menu (Window  $\rightarrow$  Preferences) you will see all the settings for Eclipse. In the window on the left is the tree with sub-nodes for navigation, above it a search box to search for nodes or settings. On the right side are the settings for the selected point.

Preferences	— 🗆 X
type filter text ×	General ⇔ ▾ ⇔ ▾ §
<ul> <li>General</li> <li>ABAP Development</li> <li>Ant</li> <li>Gradle</li> <li>Help</li> <li>Install/Update</li> <li>Java</li> <li>Language Servers</li> <li>Maven</li> <li>Model Validation</li> <li>Oomph</li> <li>Run/Debug SAP Java Connector</li> </ul>	<ul> <li>Always run in background</li> <li>Keep <u>n</u>ext/previous editor, view and perspectives dialog open</li> <li>Show heap status</li> <li>Rename resource inline if available</li> <li>Workbench save interval (in minutes): 5</li> <li>Open mode</li> <li>Double click</li> <li>Single click</li> <li>Select on hover</li> <li>Open when using arrow keys</li> <li>Note: This preference may not take effect on all views</li> </ul>
<ul> <li>&gt; Terminal</li> <li>&gt; TextMate</li> <li>&gt; Validation</li> <li>&gt; Version Control (Team)</li> <li>&gt; Web Services</li> <li>&gt; XML</li> <li>&gt; XML (Wild Web Developer)</li> </ul>	Restore <u>D</u> efaults <u>Apply</u>
? 🎽 🖌 🖲	Apply and Close Cancel

#### Figure 137 Getting Started with Global Settings



In the following sections, we'll show our suggestion for these settings, the path within the settings to find them, and a brief explanation of the effects.

#### 6.1.1.1 Dark Theme

(General  $\rightarrow$  Appearance)

Many development environments now offer the possibility to work with a light or dark theme to protect the eyes or simply to suit personal taste.

Appearance		+	•	<b>→</b> ~	:
✓ Enable theming					
Theme:	Dark				~
Color and font theme:	Default (current)				~

#### Figure 138 Setting for the Dark Theme

#### 6.1.1.2 Indentation

(General  $\rightarrow$  Editors  $\rightarrow$  Text Editors)

When writing ABAP source code, a tab width of 2 spaces is often used, but by default it is set to 4 ("Displayed tab width") in Eclipse. You can also set whether spaces are inserted instead of a tab ("Insert spaces for tabs") and whether an entire tab should be removed when deleting ("Remove multiple spaces on backspace/delete").

Preferences			—	
type filter text	Text Editors		<	<b>(-) -</b>
<ul> <li>General</li> <li>Appearance</li> <li>Compare/Patch</li> <li>Content Types</li> </ul>	Some editors may not honor all of these settings. See <u>'Colors and Fonts'</u> to configure the font.			
<ul> <li>Editors</li> <li>Autosave</li> <li>File Associations</li> </ul>	<u>U</u> ndo history size: Displayed <u>t</u> ab width: <u>E</u> nable word wrap when opening an editor	200 2		
Structured Text Editor     Text Editors	Insert spaces for tabs Remove multiple spaces on backspace/delete			

#### Figure 139 Source Code Indentation Setting



#### 6.1.1.3 Error message in the Code

### $(General \rightarrow Editors \rightarrow Text Editors)$

Error messages appear as icons to the left of the source code. To get the information about the error, you need to hover over the icon with the mouse. You can also use the option "Show code minings for problem annotations" to display the entire error message directly in the code. To do this, select the type of message.

Show code minings <u>for problem annotations</u> :	Errors, Warnings and Infos $\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!\!$			
Maximum annotations shown:	100			

#### Figure 140 Example of Setting

 ▲ Use functional writing style: 007 - Functional writing style for CALL METHOD CALL METHOD main.
 ◆ EXPORTING can be omitted: 030 - EXPORTING can be omitted | ▲ Ungelesene Felder: Erweiterte Programmprüfung · DATA(test) = one_parameter( EXPORTING i_test = abap_true ).
 ◆ Irreführender Feldname: Erweiterte Programmprüfung (SLIN) | ▲ Unbenutzte Felder: Erweiterte Programmprüfung data x TYPE i.
 ■ Found a suspicious comment: ...: 070 - Find comment markers "Fixme

Figure 141 Result Image in Source Code

#### 6.1.1.4 Spell checker: Comments

(General  $\rightarrow$  Editors  $\rightarrow$  Text Editors  $\rightarrow$  Spelling)

If you don't write comments in English in Eclipse, the spell checker will give you a lot of red comments. You can deactivate this check via the settings ("Enable spell checking") or you can download the dictionary for your language.

#### 6.1.1.5 Shortcuts

(General  $\rightarrow$  Keys)

Configuration of the keyboard shortcuts in Eclipse that allow you to define your desired settings. In addition, you can display shortcuts if they have been triggered ("Through keyboard") or if there is a key combination for the executed action ("Through mouse



click"). This option is always useful if you want to give training courses, help colleagues familiarize themselves with Eclipse or familiarize yourself with the keyboard shortcuts.

See also:

Blog Post Useful Keyboard Shortcuts for ABAP in Eclipse

SAP Help Keyboard Shortcuts for ABAP Development

6.1.1.6 Debugging

(ABAP Development  $\rightarrow$  Debug)

Possibility to specify the general debugger settings, but also to enable ("Enable debugging of system programs") of system debugging.

### 6.1.1.7 Color formatting

(ABAP Development  $\rightarrow$  Editors  $\rightarrow$  Source Code Editors  $\rightarrow$  ABAP Keyword Colors)

To highlight important keywords in Eclipse, you can highlight them with additional color combinations. To do this, you can highlight individual keywords or select all ("Select all"). This makes it easier to identify important passages in the source code.



vne filter text	ABAP Keyword Colors		
General     ABAP Development     Activation	Add keywords to assign color and font styl	le (* = any string):	Add
Cross Trace Debug Coss Trace Debug Coss Code Editors ABAP Keyword Colors ABAP Templates Access Control Templates Annotation Definition Templat	<ul> <li>endmethod</li> <li>return</li> <li>exit</li> <li>continue</li> <li>check</li> <li>select</li> <li>with</li> <li>endwith</li> </ul>	2	Remove Duplicate Select All Deselect All Move Up
Behavior Definition Templates CDS Code Completion Data Definition Templates DDL Formatter Dictionary Structure Templates Dynamic Cache Templates Element Info Entity Buffer Templates JSON Templates Mark Occurrences Metadata Extension Templates	<ul> <li>insert</li> <li>update</li> <li>modify</li> <li>testing</li> <li>test-injection</li> <li>end-test-injection</li> <li>test-seam</li> <li>end-test-seam</li> <li>end-test-seam</li> <li>message</li> <li>raise</li> <li>public</li> </ul>	2	Move Down Color: Bold Italic Strikethrough
Quick Assist Service Definition Templates Transformation Templates > Graphical Tools	protected     private	Restore Defaults	Export Import Apply

Figure 142 Color Settings to Highlight the Keywords in the Source Code

#### 6.1.1.8 Code Templates

(ABAP Development  $\rightarrow$  Editors  $\rightarrow$  Source Code Editors  $\rightarrow$  ABAP Templates)

For frequently used code fragments, SAP delivers templates that can be adapted to your own needs. New templates are also possible. The templates are inserted in the coding by entering the template name and autocomplete (CTRL+SPACE).



type filter text	Templates				⟨¬ ▼ ⊲⟩ ▼ §
> General	Create, edit or	remove templates:			
Activation	Name	Context	Description	Auto Ins	New
Debug	case	ABAP	Case block	on	Edit
Editors     Source Code Editors	<ul><li>✓ catch</li><li>✓ firePlug</li></ul>	ABAP Web Dynpro ABAP	##CATCH_ALL pr Fire Plug (use in v	on on	Remove
ABAP Keyword Colors ABAP Templates	functi	ABAP Web Dynpro ABAP	Function Module	on	Restore Removed
Access Control Templates Annotation Definition Templat	if	АВАР	If block	on	Revert to Default
Behavior Definition Templates > CDS	<ul> <li>✓ ifElse</li> <li>✓ instan</li> </ul>	ABAP Web Dynpro ABAP	If-else block Instantiated Used	on on	Import
Code Completion Data Definition Templates	✓ Icl ✓ Iif	ΑΒΑΡ ΔΒΔΡ	Local class	on	Export
DDL Formatter Dictionary Structure Templates Dynamic Cache Templates Element Info Entity Buffer Templates JSON Templates Mark Occurrences	Preview: class lcl public metho privat	_app definitio section. ds main. e section.	n create <b>publi</b>	c final.	
Metadata Extension Templates				Restore Defaults	Apply

#### Figure 143 Administration of ABAP Templates in Settings



#### Figure 144 Inserting the Template into the Source Code

# 6.1.1.9 Alias for CDS Fields

(ABAP Development  $\rightarrow$  Editors  $\rightarrow$  Source Code Editors  $\rightarrow$  CDS  $\rightarrow$  Code Completion)

When creating a Core Data Service (CDS), when a table is included, the field names are provided without an underscore and in Camel Case with an alias. With the option ("Add aliases for table fields ..."), this is done by default when inserting via "Insert all elements" (default setting).

#### 6.1.1.10 Auto-complete

# (ABAP Development $\rightarrow$ Editors $\rightarrow$ Source Code Editors $\rightarrow$ Code Completion)

By default, Eclipse adds parentheses and quotation marks at the end of an expression ("Automatically close brackets and literals") and inserts spaces inside parentheses ("Add additional whitespace inside ..."). If you are bothered by these options, they can be disabled here. Furthermore, you can also have non-keywords suggested by Eclipse ("Also suggest non-keywords"), which will then also suggest variable names, for example.

#### 6.1.1.11 Search

(ABAP Development  $\rightarrow$  Search)

Here you can make settings in the search dialog (**CTRL+SHIFT+A**), e.g. whether the old search pattern is continued ("Use pattern from previous search") or the number of hits displayed ("Maximum number of results"). However, the type of the object ("Display object types") and in which package ("Display packages") it is located is also important.

# 6.1.2 **Project-specific Settings**

You can find the system-specific settings by right-clicking on the ABAP project under "Properties". The structure of the window is similar to that of the global settings and can be operated immediately.



Properties for A4C_EN		— 🗆 X
type filter text ×	Resource	<-> ▼ -> ▼ 8
<ul> <li>&gt; Resource</li> <li>&gt; ABAP Development Builders</li> <li>Coverage</li> <li>Project Natures</li> <li>Project References</li> <li>Refactoring History</li> <li>Run/Debug Settings</li> <li>Semantic Resource</li> <li>Service Policies</li> <li>Task Tags</li> <li>&gt; Validation</li> <li>WikiText</li> </ul>	Path:       /A4C_EN         Type:       Project         Location:       semanticfs:/A4C_EN         Last modified:       1. Januar 1970, 01:00:00         Text file encoding       Inherited from container (Cp1252)         Image: Im	Restore Defaults Apply
?		Apply and Close Cancel

Figure 145 Getting Started with Project-specific Settings

#### 6.1.2.1 External Debugging

#### (ABAP Development $\rightarrow$ Debug)

With this option you can set for which user debugging is active. This can be the current user ("Logon User") or another user ("User") if you want to perform external debugging.

Breakpoints in Eclipse are automatically active for all access types (SAP GUI, ABAP Unit, HTTP, RFC). There is no explicit "external debugging" anymore.

#### 6.1.2.2 Pretty Printer

(ABAP Development  $\rightarrow$  Editors  $\rightarrow$  Source Code Editors  $\rightarrow$  ABAP Formatter)

As with Pretty Printer, you make the formatting settings here when the Code Formatter (SHIFT+F1) is executed. A standard for this would be, for example:



Indentation					
🗹 Indent Lines					
Format Functional Method Calls	Format Functional Method Calls				
Condense One-Line-Method Call					
Upper/Lower Case Conversion					
○ None					
<ul> <li>Derived from First Statement</li> </ul>					
Custom					
Keywords	Identifiers				
Upper Case	O Upper Case				
O Lower Case	Lower Case				
Keep Camel Case Identifiers					

Figure 146 Possible Settings for the Pretty Printer / ABAP Formatter

(Settings depending on the backend system version, e.g. "Keep Camel Case Identifiers" is only available in S/4HANA)

# 6.2 Views and Perspectives

Definitions of terms: see Chapter 1

Working with the different views: see chapter 4

#### 6.2.1 Views

All the information you see and work with is displayed in views, such as the Project Explorer or Editor. Views can be moved to the screen as desired by touching (holding the mouse button, "drag") and dragging the table tab of the view.



#### Figure 147 Move the View via the Label/Tab

As you move, you'll see a preview of the new layout.





Figure 148 The Markings Indicate the Placeability of the Window

After dropping, the view is moved to this point.

Views can also be placed outside the Eclipse screen and exist there. This is especially useful when working with multiple monitors.

If you drag the view next to another table tab, the views are stacked, i.e. combined in a view group.



Figure 149 Display of Stacked Views

View groups can be minimized and restored together.



Figure 150 Minimizing View Groups



The result is that the view group is minimized at the edge of the screen. You can use the "Restore" button to restore the view group.



Figure 151 Restoring the View Groups

After double-clicking on a table tab, the view is displayed in full screen. This is particularly useful for editor views or large screens that are displayed in the SAP GUI View. Double-clicking on the table tab again reduces the size of the view.

Views that are no longer needed can be closed using the close icon – for example, the Feature Explorer after you have worked through the tutorial.

🚰 Feature Explorer	×
Welcome!	Close
This interactive vie development team	w allows you to . These are fea

Figure 152 Closing a View

You can add new or accidentally closed views to a perspective at a later date.





Figure 153 Displaying a View

In this way, non-ABAP views (see "Recommended Additional Views") can also be assigned to the ABAP perspective.

# 6.2.2 Perspectives

The arrangement of all views on the screen is stored in a perspective. For different activities, separate perspectives are delivered, which can be adapted as desired.

In the ABAP development tools, the ABAP and debugging perspectives are mainly used, between which you can switch as you wish.



Figure 154 Switching Between Different Perspectives

*Tip: Especially in the first few weeks of ADT use, people tend to forget to return to the ABAP perspective after a debugging session.* 

If you have adjusted your perspective "too much", you can use the menu to restore the perspective to its delivery state.



	Wine	dow Help						
ř		New Window		g 🗸	🕅 🗉 🕯	- () -	· 🚷	- 💁 +
		Editor	>		[C11] ZMK11	[C11]	] YSTS	TFOO $ imes$
1		Appearance	>	►	VSTSTF00 >			
		Show View	>		1 REPORT yst	stf00.		-
i		Perspective	>	B	Open Perspective		>	TION C
]		Navigation	>		Customize Perspec	tive		
1	B	Working Sets	>		Save Perspective A	s		
Ĩ		Preferences			Reset Perspective			
ľ					Close Perspective	Res	et Pers	pective
-	J, DE]	]			Close All Perspectiv	ves		
۰l	J DF1		L		11			

Figure 155 Reset a Perspective

#### 6.2.2.1 Own Perspectives

You can also define your own perspectives. This is especially useful if you work with different monitor configurations (e.g. two monitors). This allows the size and arrangement of the views to be adjusted. Your own perspective can be created via "Save Perspektive As..." can be stored.

<u>W</u> indow <u>H</u> elp			
New Window		🎄 🕶 🔘 🕶 🤬 🕶 🦗 🕶 🌺 🖻	•
Editor	>	🔄 🍸 🕴 🗖 🔲 🕝 [NPL] ZCL_MY	_A
Appearance	>	► C YSTSTF00	•
Show View	>	1 REPORT	ys
Perspective	>	😰 Open Perspective > L	cl
Navigation	>	Customize Perspective	IC.
Preferences		Save Perspective As	нс
		Reset Perspective	÷
		Close Perspective	1
		Close All Perspectives 53	s.

Figure 156 Save a Perspective





Figure 157 Naming the new Perspective

		Q	i 🗈   🔼	ABAP DS	SAG
 Den Editors X		_	2 El		8
G zcl_my_amq_daemon.aclass					
vststf00.asprog					
					B
					2
					-
					1
ABAP Element Info ×					
	🔄 🛃 🗇 🗆	0   🖻	$ \hat{A}   \hat{A}$	-> <i>s</i> /	
Code information is not available					

Figure 158 New Perspective with Name

# 6.3 Recommended additional views

Views already installed via Eclipse or ABAP Development Tools:

- ABAP Element Info
- Minimap



Views that can be installed via Eclipse Marketplace

- Open Editors

# 6.4 Suggestions for distribution

Setting up a virtual workplace is as individual as setting up a real one. In addition to personal preferences, it also depends, for example, on the size of the monitor, how many views can be displayed at the same time. Therefore, only very subjective suggestions can be made here.

Project Explorer	Editor	Open Editors	
		ABAP Element Info	
Outline		Minimap	¥ 🕸 🔺

Figure 159 Possible Setting of the ABAP Perspective

Project Explorer	Editor	Variablen
		Callstack ("Debug")
	Interne Tabellen	

Figure 160 Possible Setting of the Debugger Perspective



# 7 <u>Plug-ins</u>

# 7.1 Introduction

Eclipse is an integrated development environment (IDE) that consists of many small units called plug-ins. If you look at the Eclipse variants Eclipse IDE for Java Developers or Eclipse IDE for C/C++ Developers as examples, then these preconfigured packages are only collections of plug-ins that have been developed for a specific purpose.

The ABAP Development Tools (ADT) are categorically exactly the same, i.e. a collection of plug-ins, and thus provide ABAP developers with a modern development tool. It is precisely this modular structure that now enables every developer to create their own plug-ins in order to further adapt Eclipse and/or ADT and thus, for example, simplify recurring tasks or provide functions that are not offered by ADT and can only be found in the ABAP Workbench.

Such plug-ins can consist of pure UI code, such as the ABAP Favorites plug-in, which brings functions of the Easy Access Menu (SAP GUI) to Eclipse. However, they can also be more extensive and require extra ABAP code on the SAP system. An example of this is the ABAP Code Search plug-in, which is comparable to the SAP GUI transaction CODE_SCANNER.

# 7.2 Useful Open-Source Plug-ins

# 7.2.1 Open Editors

Provides a new view that shows all open editors in Eclipse. This view also offers the possibility to adjust the sort order of the editors.



Sort Sequence for Open Editors			×
Use Drag and Drop to reorder the sort sequence.			
File Name			
File Path			
File Extension			
Tab Order in Eclipse			
Last Access Time			
OK	(	Car	icel

Figure 161 Dialog for Adjusting the Sort Order of Open Editors

Prerequisites Eclipse:

• Eclipse IDE for Java Developers

Links:

- Source-Code on GitHub
- Eclipse Marketplace

#### 7.2.2 AnyEdit Tools

Offers many options for editing text/source code:

- Conversion of text to lowercase/uppercase letters
- Conversion of text from Pascal to Camel notation
- Sorting of the selected rows (alphabetically, numerically, by line length)
- ...

In addition, the plug-in also offers many options for comparing text.

- Comparing an editor to text on the clipboard
- Comparing an editor with any other editor
- Comparing an editor with an external file





All possible operations are available via the context menu of an editor.

Figure 162 Examples of Available Operations in the Context Menu

#### Prerequisites Eclipse:

• Eclipse IDE for Java Developers

#### Links

- Source-Code on GitHub
- Eclipse Marketplace
- ADT in Eclipse As of March 2023



# 7.2.3 PDE Tools

This plug-in extends Eclipse with useful tools for plug-in developers:

- Preview icon files directly in Project Explorer
- Generation of Java constants for icon folders
- Screenshot tool for UI elements in Eclipse Workbench

However, there are features that are also helpful outside of plug-in development:

- Extended clipboard history
- Direct launch of a new Eclipse window with a specific workspace



Figure 163 Clipboard History (Keyboard Shortcut Ctrl+Shift+V)

Links:

- Source-Code on GitHub
- Eclipse Marketplace



# 7.3 Useful Open-Source ADT Plug-ins

# 7.3.1 ABAP Favorites

The ABAP Favorites plug-in was developed to map the functionality of the SAP GUI User Menu. In this menu, each user can add transactions, reports or URLs to their favorites and structure them according to their personal preferences.

The plug-in installation brings two new views, Favorites and Favorite DevObjects, available via Windows  $\rightarrow$  Show View  $\rightarrow$  Others. Both views offer a filtered tree view in which the favorite objects can be managed.

The difference between the two views lies in the possibilities for creating the folders (containers). The Favorites View allows two types of these folders: "Standard" for transactions, reports and URL and "DevObject" for managing development objects such as classes, function modules, CDS views, etc. In the case of Favorite DevObjects, only "DevObject" folders can be created. The division of these views allows the user to choose whether he wants to mix all folders or manage them according to the separation described.

If you compare the "Standard" folder with the "DevObject" folder, apart from the selectable object types, the biggest difference is that with "Standard" the objects are executed by double-clicking. Double-clicking in a DevObject folder opens the selected object.



Plug-ins

\star Favorites 🗙		💠 🟠 🗘 🗘 😫	000	' 🗆
type filter text				
Name	Description	Linked To		
V 📂 ABAP	Workbench			
> 늘 ABAP Enhancements	Enhancements			
> 늘 ABAP Performance	Performance			
> 늘 ABAP Tools				
> 늘 ABAP Transports	Transports			
🗸 늘 CDS Views				
RUTDDLSACT	Activates Set of DDL Sources			
RUTDDLSANALYZE	Analysis of DDL Sources			
RUTDDLSDEL	Delete the DDL Source and the generated view			
RUTDDLSSHOW	Search for DDL sources with name and/or temp	olate		
RUTDDLSSHOW2	CDS Source Display			
> 늘 Dictionary Objects				
> 늘 Error monitor	Log-Points			
> 늘 JOBS	ABAP JOBS			
> 늘 Printouts				
🗸 늘 SPRO Menu	Extension of SPRO			
S_CUS_IMG_ACTIVITY	Edit Activity			
S_IMG_EXTENSION	Edit IMG/SPRO Structure			
SIMGH	Edit IMG Structure			
SLICENSE	SAP* Appl1ance			
> 📂 Common	Common T-Codes			
> 늘 Fiori	Fiori configuration			
> 늘 HANA Convertion				
> 📂 IDOCS	IDOCs EDI			
> 📂 Personas	SAP Screen Personas			
> 📂 RFC				
> 늘 WebGUI	Links			

Figure 164 ABAP Favorites View

To add new objects to the favorites, you can use the context menus of the Favorites View, the ABAP Editor, or the Project Explorer.



🗸 📂 ABAP				Workbench
> 늘 ABAP Enhancem		Enhancemen		
> 📂 ABAP Performa	4	Add New Folder		Performance
> 늘 ABAP Tools	÷	Add Transaction		
🔉 🏷 ABAP Transport		Add Iransaction		Transports
🗸 📂 CDS Views 🛛	Р	Add Program	Tran	saction
P RUTDDLSAC	U	Add URL		Activates Set
RUTDDLSAN			_	Analysis of D
RUTDDLSDE	×	Delete Folder		Delete the DI
RUTDDLSSH	2	Edit		Search for DI
RUTDDLSSH	_			CDS Source [
> 늘 Dictionary Obje	台	Go Home		
> 늘 Error monitor 🕔	Ģ	Go Back		Log-Points
> 📂 JOBS 💡	4	Go Into		ABAP JOBS
> 📂 Printouts 👘				

Figure 165 Context Menu of a Folder in ABAP Favorites View

Prerequisites:

- Eclipse IDE for Java Developers
- ADT

Links:

- Source-Code on GitHub
- Eclipse Marketplace

#### 7.3.2 ABAP Continuous Integration

AbapCI is an open source Eclipse plug-in that provides various Continuous Integration (CI) tools for ABAP development with Eclipse. The plug-in is based on ADT's CI capabilities.

The plug-in provides the following functions:

- Automatic unit test runs
- Automatic ATC runs
- Visualization of the source code status on the user interface
- Different color schemes for each ABAP project
- Automatic source code formatting
- Shortcut for abapGit



<ul> <li>Triggering Jenkins from Eclipse (experimental)</li> </ul>							
	CI run package summary ι						
17:55:04:							

Figure 166 Colored Highlighting of the Status Bar per Project + Test Status

🗢 ABAP Continuous Integration 🗙 🗢 ABAP Colored Projects 🗢 ABAP CI Dashboard 🕂 🖶 🖉 🗖 🗖														
Project name	Package name	Unit tests	#	Err	Sup	Last run	ATC state	#	Err	Warn	Info	Sup	Last run	First error
	TEST_CALL_HIER	NOK	26	1	0	17:55	DEACT							ZCL ACALLH ABAP I

#### Figure 167 Management of Packages for which Unit Tests and/or ATC Test Runs are Scheduled

More information can be found in the GitHub repository.

Prerequisites:

- Eclipse IDE for Java Developers ( <= 2022-06, installation with newer versions currently only possible with workaround; see issue on GitHub)
- ADT

Links:

- Source-Code on GitHub
- Eclipse Marketplace

# 7.3.3 ABAP ADT Extensions

This plug-in extends ADT with several additional functionalities.

#### 7.3.3.1 Automatic login to SAP systems

The "Automatic Login" functionality allows the developer to manage their user/password combinations within Java's Secure Storage. From a security point of view, these should be encrypted.



Passwords ×		- 8
type filter text		
Project/Client/User	Password	Encrypt
✓ ☑ 🗁		false
✓ ☑ ➢ 100		false
	******	true
✓ ☑ 🗁		false
✓ ☑ ➢ 100		false
	******	true
✓ ☑ 🗁		false
✓ ☑ ➢ 100		false
	******	true
✓ ☑ 🗁		false
✓ ☑ ➢ 100		false
	******	true
✓ Ø → Ø → Ø → 100	****	false false true

Figure 168 View for Managing the Stored Access Aata of ABAP Systems

Depending on the settings of the plug-in, you can automatically log in to any of the selected on-premise SAP systems that are mapped using ABAP/BW projects. The passwords can be maintained when the project is created or later via the password view.

# 7.3.3.2 Changing ABAP Project Attributes

The context menu of the Project Explorer can be used to change the project attributes (client, user, language) for ABAP/BW projects. In addition, the breakpoint users can be set.


Q.	Coverage As	>	
≁	Profile As	>	
	Coverage As	>	
0	Run As	>	
*	Debug As	>	
₿	Profile As	>	
	Restore from Local History		
78	SQL Trace		
Ċ.	SQL Console		
	ABAP Project	>	Logon with Secure Storage
	PDE-Tools Shell	>	Logon to All SAP Systems
	Team	>	Change Logon Language
	Compare With	>	Change Logon User
	Configure	>	Change Logon Client
	Source	>	Set breakpoint user to logon user
	Validate		Select user for breakpoints
_			

Figure 169 Context Menu on Project in Project Explorer

### 7.3.3.3 Input field for executing transaction codes

After installation, a toolbar appears in the lower right area of Eclipse with an input field for transaction codes. The field can be operated by mouse or via the shortcut **Shift+F8**. After entering the code and confirming with Enter, the transaction is executed in the currently active project.



Figure 170 Status Bar in the Eclipse Window

Prerequisites:

- Eclipse IDE for Java Developers
- ADT

Links:

- Source-Code on GitHub
- Eclipse Marketplace



### 7.3.4 ADT Classic Outline

This plug-in adds a new view called "Classic Outline" to your interface, which in a sense maps the SE80 object list. In most cases, this view can replace the built-in ADT outline. The displayed object list can be filtered and allows the object attributes to be viewed, similar to the built-in ADT outline. By double-clicking (or single clicking depending on the settings) you can navigate to the selected object.

E Classic Outline ×	S	
type filter text		
V 🕝 ZCL_ABAPGIT_AJSON (66) AJSON		^
> 🚺 Interfaces (1)		
> 🔺 Attribute (9)		
✓ ● Methoden (34)		
> 1 ZIF_ABAPGIT_AJSON (28) AJSON interface and types		
CREATE_EMPTY		
CREATE_FROM		
e PARSE		
DELETE_SUBTREE		
GET_ITEM		
PROVE_PATH_EXISTS		
> 🛞 Typen (5)		
✓   ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓		
> 6 LCL_ABAP_TO_JSON (30) final		
V 🚱 LCL_FILTER_RUNNER (7) final		
✓ ● Methodendefinitionen (2)		
RUN		
WALK		
Methodenimplementierungen (2)		
RUN     NALK		
WALK		
GET LCL ISON DARSER (12) final		
S G LCL ISON SERIALIZER (12) final		
S G LCL ISON TO ABAP (18) final		
S G LCL NODES HELPER (7) final		
S G ICL UTILS (6) final		
S I LTCL ABAP TO JSON (25) for testing: final		
S ILTCL FILTER TEST (10) for testing: final		

#### Figure 171 Classic Outline View

Prerequisites:

- Eclipse IDE for Java Developers
- ADT



Prerequisites ABAP:

- SAP NetWeaver 7.40 SP08 or newer
- abapGit repository ADT Classic Outline Backend must be installed

### Links:

- Source-Code on GitHub: Frontend and Backend
- Eclipse Marketplace

### 7.3.5 ABAP Quick Fix

Quick fixes are part of the Eclipse IDE. In the ADT standard, they are processed in the backend system and can be used by the user if required using the shortcut **CTRL+1**. The ABAP Quick Fix plug-in provides additional quick fixes that are processed directly from the Eclipse environment.



Figure 172 ABAP Code Before Quick-Fix Execution

794	lo_to_abap	
795	= NEW #( ii_custom_mapping = mi_custom_mapping	).

#### Figure 173 ABAP Code After Quick-Fix Execution

For an excerpt of the available features, see the following list:

- REplace READ TABLE with ASSIGN, REF#, Table Expression or line_exists.
- Replace CALL METHOD with the direct call.
- Replace MOVE with direct assignment.



- Change APPEND TO to APPEND VALUE#() TO.
- Replace CREATE OBJECT with NEW.
- Replace "full line comments" of statement.
- Omit the self-reference ME->.
- Replace the operators EQ, NE, GT, GE, LT, LE with =, <>, >, >=, <, <=
- Indent operators in the selected area accordingly.
- Indent TYPE and LIKE in the declaration block corresponding to the variable.

### Prerequisites:

- Eclipse IDE for Java Developers
- ADT

### Links:

- Source-Code on GitHub
- Eclipse Marketplace

### 7.3.6 ABAPQuickFixS4Conversion

This plug-in is a very nice example of developer collaboration within the community. ABAPQuickFixS4Conversion is an extension of the ABAP Quick Fix Plug-in from SAP and adds the following functionalities:

- Convert SELECT SINGLE to SELECT ... UP TO 1 ROWS ... ORDER BY
- Adjust the custom ORDER BY list for any table
- Change SELECT SINGLE to modern SQL style
- Convert SELECT/ENDSELECT to modern SQL style
- Transform MOVE_CORRESPONDING to CORRESPONDING #()



38	
39	select single tkonn tposn from wbit into (tkonn, tposn)
40	where to FRemove all ABAP Comments
41	and t or Paplace select single with select up to one rows
42	Replace select single with select up to one lows
43	
44	
45	
46	

Figure 174 Example of Quick Fix Availability on a SELECT Statement

38	
39	select tkonn, tposn
40	from wbit
41	into (@tkonn,@tposn)
42	up to 1 roks
43	where tkonn = @tkonn
44	and tposn = @tposn
45	order by doc_type, doc_nr, doc_year, item, sub_item.
46	endselect.
47	

Figure 175 SELECT Statement after Applying the Quick Fix

Prerequisites:

- Eclipse IDE for Java Developers
- ADT
- ABAP Quick Fixes plugin

Links:

- Source-Code on GitHub
- Eclipse Marketplace



### 7.3.7 ABAP Tags

The ABAP Tags plug-in allows you to create tags that can then be added to any development object. The tags and their assigned objects are persisted on the respective ABAP system. This makes it easier for other users to access tagged objects. In general, the plug-in allows you to create tags either in the global or in the user-specific scope. User-specific tags can also be shared with others, making collaboration easier.



Figure 176 View "Tag Manager"

Tagging objects is intuitively possible via the context menu from the editor or the Project Explorer. The tagged objects can then be searched for and displayed either from the "Tag Manager" view using the context menu action, or via the "ABAP Tagged Object Search" integrated in the "Search" dialog.



Figure 177 Search-Dialog with "ABAP Object Search"

Prerequisites Eclipse:

- Eclipse Platform Runtime or Eclipse IDE for Java Developers
- ADT

Prerequisites ABAP:

- SAP NetWeaver 7.40 SP08 or newer
- abapGit repository abap-tags-backend must be installed

Links:

- Source-Code on GitHub
- Eclipse Marketplace (Marketplace Client must be installed)



### 7.3.8 ABAP Search and Analysis Tools

This plug-in extends ADT with additional search and analysis capabilities for the following object types:

- Class/Interface
- Database table/view
- CDS View

The search functions are integrated into the Eclipse search dialog (Ctrl+H). The operation is similar to the "Open ABAP Development Object" dialog (Ctrl+Shift+A). The object type can be changed using a dropdown. Among other things, this controls the available filters in the "Search Filters" field.

Cearch		_	×	
ABAP Object Search (DevEpos)				
<u>T</u> ype: CI	OS View 🗸			
Object <u>N</u> ame:				
Search <u>F</u> ilters:				
Use "AND" instead of "OR" dur	🖞 owner			Use 'owner' to restrict the search query by specific
Maximum number of results:	🖶 package 🏟 desc			Example:
Project: *	api from assoc field			owner : smith
(?) Customi <u>z</u> e	<ul> <li>netu</li> <li>extby</li> <li>anno</li> <li>param</li> <li>params</li> <li>type</li> </ul>			

Figure 178 Search Dialog on Page "ABAP Object Search"

In addition to the search functions, the plug-in also provides the "CDS Analyzer" view, which allows the following analyses on CDS views:

- Top-Down
- Evaluation of all used entities of a CDS View
- Where-used list of database entities as data source ("select from" or "association")



- Field-level analysis
  - Top-down (origin determination)
  - Bottom-up (use of a field in fields of other CDS views)



Figure 179 View "CDS Analyzer" - Top-Down-Analysis

Prerequisites Eclipse:

- Eclipse Platform Runtime or Eclipse IDE for Java Developers
- ADT

Prerequisites ABAP:

- SAP NetWeaver 7.40 SP08 or newer
- abapGit repository abap-search-tools must be installed

Links:

- Source-Code on GitHub
- Eclipse Marketplace (Marketplace Client must be installed)

#### 7.3.9 ABAP Code Search

This plug-in brings the well-known SAP GUI transaction CODE_SCANNER to Eclipse. As the name suggests, the "ABAP Code Search" is also integrated into the Eclipse Search dialog. In addition to the use of regular expressions, there are also special



search modes such as: *Single Pattern mode* or *Sequential Matching*. Other features of the search are:

- Parallel execution (optionally controllable per user)
- Search can be stopped at any time
- Complete system search possible, since only small packets are processed on the application server at a time
- ABAP Tags plug-in Tags can be used for object selection → requires installation of the ABAP Tags plug-in

Search Pattern: *			
^		⊡ Ignore □ <u>R</u> egula	<u>C</u> ase ir Expressions
¢	>		
Object Selection		Additional	Settings
Dbject Name:		Ignore o	comment lines
		Multilin	e search
jilters:		Match a	attern mode
		Sequent	tial Matching
Tags:			
💊 SQL 🗴 🔖 OData 🗴	2000 (Aliante de la constante		
Class includes to search All O Selected ( <u>0 of 8 selected</u> )	Function Group in	ncludes to se I ( <u>0 of 2 sele</u>	arch cted)
roject: * A4H_001_developer_en			Browse

#### Figure 180 Search-Dialog with "ABAP Code Search"

#### Prerequisites Eclipse:

- Eclipse Platform Runtime or Eclipse IDE for Java Developers
- ADT



Prerequisites ABAP:

- SAP NetWeaver 7.40 SP08 or newer
- abapGit repository abap-code-search-tools must be installed

### Links:

- Source-Code on GitHub
- Eclipse Marketplace (Marketplace Client must be installed)

### 7.3.10 abapGit Eclipse Plug-in

Provides the functions of the abapGit SAP GUI transaction as an Eclipse plug-in. However, the full range of functions is currently (2022) only guaranteed via the SAP GUI transaction.

Prerequisites Eclipse:

- Eclipse IDE for Java Developers
- ADT

Prerequisites ABAP:

- SAP NetWeaver 7.50 or newer
- Complete abapGit_Installation
- abapGit repository ADT_Backend must be installed

### Links:

- Source-Code on GitHub
- Installation via Update-Site https://eclipse.abapgit.org/updatesite/

## 7.4 Develop own ADT Plug-ins

### 7.4.1 Prerequisites

Since plug-ins for Eclipse are to be developed in Java, it is advisable to already have some knowledge there. The necessary level of knowledge depends on the plug-in you want to develop.



### 7.4.2 Setting up the development environment

#### 7.4.2.1 Installation Eclipse for RCP/RAP Development

To develop plug-ins for Eclipse, you need a specific variant of the Eclipse platform: Eclipse *IDE for RCP and RAP Developers* (RAP = Remote Application Platform). This variant provides a complete toolset to develop plug-ins for Eclipse as well as rich client applications (RCP) based on Eclipse. It can be obtained directly from eclipse.org.

## Eclipse IDE for RCP and RAP Developers

425 MB 10,137 DOWNLOADS

A complete set of tools for developers who want to create Eclipse plug-ins, Rich Client Applications or Remote Application Platform (RCP+RAP), plus Maven and Gradle tooling, and an XML editor. It contains the EGit tooling for accessing Git version control systems, and Eclipse Passage which helps with license management for Eclipse-based products.

Figure 181 Eclipse Bundle "Eclipse IDE for RCP and RAP Developers"

Furthermore, an installation of the Java Development Kit (JDK) is required. This can be obtained, for example, from the following sources:

- https://adoptium.net/de/
- https://openjdk.org/
- https://sap.github.io/SapMachine/

**Note**: In the more recent Eclipse versions, this is already included.

Which Java Version?

Since Eclipse v2020-09 or ADT v3.16, Java 11 is the minimum requirement, and therefore the JDK should also be installed in at least this version.

#### 7.4.2.2 Installation of ADT

The next step is to install ADT in Eclipse, otherwise it will not be possible to develop against the ADT SDK.



In order to have the best possible compatibility, it is recommended to always install ADT in the same version as Eclipse. The latest ADT version can be obtained from https://tools.hana.ondemand.com/latest. For older versions of ADT, you simply have to replace the *latest* in the path with the desired Eclipse version (for Eclipse 2020-09, for example, it would be: https://tools.hana.ondemand.com/2020-09).

#### 7.4.2.3 Installation of useful Plug-ins (optional)

In addition to the Eclipse installation, it is recommended to install the following plugins:

#### ENHANCED CLASS DECOMPILER

Marketplace Link: https://marketplace.eclipse.org/content/enhanced-classdecompiler.

This plug-in allows compiled source code to be displayed in a readable way. It is even possible to set breakpoints in such de-compiled classes and parse the code at runtime.

#### WEB DEVELOPER TOOLS

Marketplace Link: https://marketplace.eclipse.org/content/eclipse-web-developer-tools-0/promo.

If you also want to offer help for your plug-in, this plug-in extends Eclipse with editors with syntax highlighting for the typical web file extensions (css, html etc.).

#### **WINDOWBUILDER**

Marketplace Link: https://www.eclipse.org/windowbuilder/

Creating GUI elements, such as dialogs or custom views, can sometimes be very timeconsuming. The WindowBuilder can help with this and allows you to create GUI elements with the help of a graphical editor.



### 7.4.3 Key Concepts/Artifacts

#### 7.4.3.1 Plug-in

A plug-in is used to aggregate code into a modular, extensible, and shareable unit. The entire Eclipse application consists of many such plug-ins.

#### 7.4.3.2 Feature

A feature is used to group one or more plug-ins into a single installable and updatable unit.

#### 7.4.3.3 Update Site

Update sites are used to organize and export features so that they can be installed in Eclipse applications.

### 7.4.4 Creation of a Plug-in Project

A new plug-in project can be created via File  $\rightarrow$  New  $\rightarrow$  Plug-in Project. The Project Wizard plug-in will open:



#### Plug-ins

🛑 New Plug-in Projec	t	 o x
Plug-in Project		
Create a new plug-in	project	
Project name: com.o	:ompany.myplugin	
Use default location	on	
Location:	And the second second second	Browse
Choose file	system: default $\sim$	
Project Settings		
🗹 Create a Java proje	ect	
Source folder:	src	
Output folder:	bin	
Target Platform		
This plug-in is targete	ed to run with:	
Eclipse		
🔿 an OSGi fran	nework: Equinox 🗸	

Figure 182 Plug-in Project Wizard – Entrance

Here, a name for the project must first be assigned. In the case of the name, the so-<br/>called.Reversenamingdomainnotationisrecommended(e.g.com.company.myplugin), but any other naming convention can be chosen here.

By default, a plug-in project is always created as a Java project because most plugins contribute code. However, this option can also be deselected, e.g. for plug-ins that only provide documentation.

Since this guide explicitly discusses plug-in development for Eclipse, the *target platform* is always Eclipse.

Clicking on *Next* takes you to the next page of the wizard.



New Plug-in Project		-			Х
ontent				-	
inter the data required to	generate the plug-in.				
Properties					
ID:	com.company.myplugin				
Version:	1.0.0.qualifier				
Name:	Myplugin				
Vendor:	COMPANY				~
Execution environment:	JavaSE-11	~	Envi	ronmen	its
Options					
Generate an activator					
Activator: com.com	pany.myplugin.Activator				
🗹 This plug-in will make	contributions to the UI				
Enable API analysis					
Rich Client Application					
Create a rich client applic	ation?	0	) Yes	N	0

Figure 183 Plug-in Project Wizard – Content

This is where the plug-in-specific properties are recorded. It is recommended to use the project name for the ID, but this is not mandatory. The version must follow the pattern *major.minor.micro.qualifier*. The *.qualifier* part is optional. It is replaced with a timestamp during the build (e.g. 1.3.0.202205011550).

The *Name* and *Vendor* fields are translatable and represent the plug-in name and its vendor.

For the *Execution Environment*, the minimum required Java version must be entered. Java 11 is currently the minimum requirement for ADT plug-ins and should therefore also be set for your own plug-ins.

If the *Generate an activator option is set,* an *activator* class is generated. Such a class can exist exactly once per plug-in and is only necessary if activities are necessary when starting or stopping the plug-in. The option *This plug-in will make contributions to the UI* regulates the available templates, which can be selected on the next page of the wizard.

As a last optional step, a template can be selected on the next page. Templates exist, for example, for creating your own views or editors.



After completing the wizard with *Finish*, the plug-in project is generated at the selected location in the file system and then displayed in the Eclipse Workspace.

#### 7.4.4.1 Structure of a Plug-In Project

A plug-in project always has the following structure. The plugin*.xml* file and the OSGI-INF folder are optional and only exist if necessary.



Figure 184 Plug-in-Projekt in the Project Explorer View

The most important files are *manifest.mf*, build.properties, and *plugin.xml*. If you open one of these three files, the plug-in manifest editor is opened by default. This editor enables the maintenance of all metadata of a plug-in, which the editor subdivides into the following areas:

#### - Overview

This view serves as a general entry point. It allows the maintenance of basic plug-in data such as name, version, etc. and the jump to the other views.

#### - Dependencies

All plug-ins that are required in this plug-in must be listed here.

#### - Runtime

Used to specify the Java packages that should be visible to other plugins.

An API status can also be set for each package.



### - Extensions

This is where the actual extension of Eclipse takes place, e.g. with new menus, commands, views, etc.

### - Extension Points

Definition of the extensibility points that this plug-in provides to others.

- Build

Configuration of which files should be included in the build result.

### 7.4.5 Creation of a Feature Project

A new feature can be created via File  $\rightarrow$  New  $\rightarrow$  Feature Project. The Feature Project Wizard opens:

New Feature				×			
Feature Properties         Define properties that will be placed in the feature.xml file							
Project name: com.co	mpany.myfeature						
Use <u>d</u> efault location							
Location: Browse							
Choose file system: default $\sim$							
Feature properties							
Feature <u>I</u> D:	com.company.myfeature						
Feature Na <u>m</u> e:	Myfeature						
Feature Version:	1.0.0.qualifier						
Feature Vendo <u>r</u> :	COMPANY			~			
Ins <u>t</u> all Handler Library:							

#### Figure 185 Feature Project Wizard - Entrance

The properties of a feature project are similar to those of a plug-in project, so the same rules apply to e.g. *ID, name,* or *version*.

Now the wizard can be closed or you can navigate to the next page, where you can select the plug-ins that should be included in this feature:



Referenced Plug-ins and Fragments Select the plug-ins and fragments from your workspace to package in new feature.	to the		<b>D</b> =
Select the plug-ins and fragments from your workspace to package in new feature.	to the		
O Initialize from a launch configuration:			
,			
Eclipse Application			$\sim$
Initialize from the plug-ins list:			
type filter text		Select Al	I
□ biz.aQute.bnd.util (6.0.0.202110041835) ↓ biz.aQute.bndlib (6.0.0.202110041835)	^	Deselect A	AII
Ch.qos.logback.classic (1.1.2.v20171220-1825)		Select	
□ ♣ ch.qos.logback.core (1.1.2.v20160208-0839) □ ✤ ch.qos.logback.slf4j (1.1.2.v20160301-0943)		Deselect	t
com.devepos.adt.abaptags.backend (1.4.6.qualifier) com.devepos.adt.abaptags.model (1.4.6.qualifier)			
Comparison of the second state of the secon	×		
0 of 1306 selected.			

#### Figure 186 Feature Project Wizard - Plug-in Selection

Upon completion, the feature project is generated and displayed in the workbench.

#### 7.4.5.1 Structure of a Feature Project

A feature project has a very simple structure. It contains only the two files *feature.xml* and *build.properties.* As with the plug-in project, there is a separate manifest editor for maintaining the feature metadata, which opens automatically when one of the two files is opened.

This is divided into the following sections:

#### - Overview

This view serves as a general entry point. It allows the maintenance of basic plug-in data such as name, version, etc. and the jump to the other views.



### - Information

Description, copyright notice and license agreement can be maintained here.

### - Included Plug-ins

Selection of plug-ins that are included in this feature.

### - Included Features

Features can be used to group other features and thus the included features can be listed here.

### - Dependencies

Normally, all dependencies are computed during the build. This is done by analyzing the dependencies of all included plug-ins. However, it is also possible to maintain the dependencies manually here.

### - Build

see Plug-in Manifest

### 7.4.6 Creation of an Update Site

A new update site project can be created via File  $\rightarrow$  New  $\rightarrow$  Project...  $\rightarrow$  Plug-in Development  $\rightarrow$ > Update Site Project. The Update Site Project Wizard opens:



Plug-ins

New Update Site				×
Update Site P	oject			
Create a new update site project				
Project name:	update-site			
🗹 Use defaul	t location			
Location: C:\	Users\stockbal\Developer\eclipse-workspaces\adt-add	-di	Browse	
Cho	ose file system: default $\sim$			
Web Resource	15			
Generate a	web page listing all available features within the site			
Web resource	location: web			

#### Figure 187 Update Site Wizard

The wizard contains only one page where you can give the project a name and select the storage location. After completing the wizard, there should be a folder in the workspace with the selected project name. This folder contains the update site's manifest file, called *site .xml*.

Double-clicking this file opens the manifest editor for the update site. In this editor, you can now add the features that you want to publish to the site. For a better overview, the features should be divided into categories.

After the content of the update site has been configured, it can be created using the *Build all* button in the editor. It is also possible to create only individual or selected features.

**IMPORTANT**: Before you create the update site, you should check the Java compiler settings again via Window  $\rightarrow$  Preferences  $\rightarrow$  Java  $\rightarrow$  Compiler. These should be set to the same Java version that has been defined as the minimum requirement for the plug-ins:



Configure Project Specific Setting
11 ~

Figure 188	Compiler	Settings	in the	Eclipse	Settings	Dialog
------------	----------	----------	--------	---------	----------	--------

If the creation is successful, the following files/folders will be located in the project folder of the update site:

- features (contains jar files of the features)
- plugins (contains jar files of the plug-ins)
- artifacts.jar
- content.jar

In addition, an archive with the name *logs.zip* may have been created. This is where all the messages that occurred during compilation are located.

#### 7.4.6.1 Testing the Update Site

Before the created site is uploaded to a web server, you may want to test it first. To do this, you should get a new Eclipse installation. The Eclipse *IDE for Java Developers* variant is completely sufficient for this. In this installation, ADT are now installed, after a restart you now add the new – so far still local – update site via Help  $\rightarrow$  Install New Software...  $\rightarrow$  Add...:

CAD C	Repository —		
Name:	Meine Plugins	Local	
Location:	file:/	Archive	
ОК			
?	Add	Cancel	

#### Figure 189 Dialog for Adding an Update Site



The "Local..." button is used to select the directory of the update site project. After clicking on the "Add" button, the categories and the assigned features of the update site should be listed and ready to be installed.

#### 7.4.6.2 Deployment

If the test of the update site is successful, it can now be uploaded to a web server to make the artifacts available to others. If you shy away from the cost of your own web server, there is a free way to provide your update site via GitHub Pages, for example. To do this, initialize a new Git repository in the project directory of the update site and publish it in a public repository on GitHub. Afterwards, the option "GitHub Pages" can be activated in the repository settings on GitHub:

Code and automation			
₽ Branches	Build and deployment		
🟷 Tags	Source		
<ul> <li>Actions</li> </ul>	Deploy from a branch -		
🔏 Webhooks			
Environments	Branch Your GitHub Pages site is currently being built from the main branch, Learn more,		
Codespaces			
💾 Pages	save Save		
	Learn now to add a Jekyli theme to your site.		

#### Figure 190 GitHub Repository Settings for GitHub Pages

After GitHub has finished creating the GitHub Pages page, it can be registered with the URL *https://<username>.github.io/<repository-name>* in Eclipse as an update site

#### 7.4.7 Extension of the ADT Backends with ABAP Code

For communication from Eclipse to the ABAP Server, ADT uses RESTful APIs. How to develop such APIs yourself can be found in the SAP Guide How To... Create RESTful APIs and consume them in ADT.

Although we are talking about RESTful APIs here, there is no communication via HTTP from ADT. It is done using the RFC protocol. At the lowest level of the communication layer of ADT, an RFC call takes place that calls a specific RFC function module on the ABAP server.

Thus, it is open to everyone to either develop their own RESTful APIs with the BAdI extension concept, as described in the guide, or alternatively to develop an RFC-



capable function module and call it with the RFC Java Connector. The Java Connector API can be integrated and used via the "com.sap.conn.jco" plug-in.

Especially for smaller plug-ins, the BAdI method may seem like quite an overhead, both on the ABAP and Java side. However, the BAdI approach also has its advantages. Especially through the use of EMF (Eclipse Modelling Framework), it is possible to easily convert a string serialized from ABAP to XML into objects in Java. Simply described, you need a "Simple Transformation" (object type XSLT) on the ABAP page to transform ABAP data to XML.

```
1 <?sap.transform simple?>
20 <tt:transform xmlns:tt="http://www.sap.com/transformation-templates"
               xmlns:cst="http://www.devepos.com/adt/cst"
               xmlns:adtbase="http://www.devepos.com/adt/base
4
               xmlns:adtcore="http://www.sap.com/adt/core"
5
               xmlns:cl="http://www.sap.com/abapxml/types/class-pool/ZIF ADCOSET TY ADT TYPES">
6
7
8
    <tt:root name="root" type="cl:ty_code_search_result"/>
9
10
    <tt:include name="sadt object reference" template="objectReferenceAttributes"/>
     <tt:include name="sadt_main_object" template="main_object"/>
11
12
130
    <tt:template>
      <tt:apply name="codeSearchResult">
149
        <tt:with-root name="code_search_result" ref="root"/>
15
16
      </tt:apply>
17
    </tt:template>
18
19
20
    <!-- Template for the result -->
21 <tt:template name="codeSearchResult">
      <tt:context>
220
23
        <tt:root name="code_search_result"/>
24
      </tt:context>
25
     <cst:result tt:extensible="deep">
26
       27
28
29
30
        <tt:attribute name="cst:queryTimeInMs" value-ref="code_search_result.query_time_in_ms"/>
31
32
        <tt:apply name="codeSearchObjects">
330
          <tt:with-root name="code_search_objects" ref="code_search_result.code_search_objects"/>
34
        </tt:apply>
35
```

Figure 191 Example of a Simple Transformation for the Transformation of ABAP <-> XML

And on the part of Java, an EMF model is necessary.



#### Plug-ins

- platform:/resource/com.devepos.adt.abaptags.model/model/abaptags.ecore
  - 🗸 🖶 abaptags
    - 🗸 📄 TagBase
      - > 🜆 ExtendedMetaData
      - > 😐 id : EString
      - > 🗖 name : EString
      - > 🗆 owner : EString
    - > 🗏 Tag -> TagBase
    - > 🗏 TagList
    - - > 🜆 ExtendedMetaData
      - > = parentObjectName : EString
      - > 🗖 parentObjectType : EString
      - > 😑 parentObjectUri : EString
        - parentTagld : EString
        - parentTagName : EString
        - 😐 image : Image
    - > 🗧 TagPreviewInfo
    - > 🗧 TaggedObject
    - > 🗧 TaggedObjectList

    - > 🖀 TagSearchScope
    - > 🖀 TagQueryType
    - > 🖀 TagInfoType
    - > 🖀 TagQueryFocus
      - Image [org.eclipse.swt.graphics.Image]

#### Figure 192 Sample EMF Model for Serializing XML <-> Java Object

This approach allows the data transformation to be built very generically, and you also save development time.

#### 7.4.8 Java Code Snippets for recurring tasks in ADT

7.4.8.1 Calling an RFC function module with the Java Connector

// 1) Reading the Destination Id for an ABAP Project Instance

String destinationId =
AdtCoreProjectService().getDestinationId(project);

// 2) Reading the JCo destination to the Destination Id

JCoDestination destination = JCoDestinationManager.getDestination(destinationId);

// 3) Reading the RFC block



```
JCoFunction function =
destination.getRepository().getFunction("name_of_rfc_function");
// 4) Setting an Importing Parameter
function.getImportParameterList().getField("I_PARAM1").setValue("PARAM_VALUE");
// 5) Executing the function
function.execute(destination);
// 6) Reading an Exporting Table Parameter
JCoTable objectTree = function.getExportParameterList().getTable("E PARAM1");
```

7.4.8.2 Reading the ABAP project depending on the current selection in the workbench

// 1) Determination of the active page in the workbench

IWorkbenchPage page =
PlatformUI.getWorkbench().getActiveWorkbenchWindow().getActivePage();

// 2) Reading the Workbench Window Instance

IWorkbenchWindow window = page.getWorkbenchWindow();

// 3) Determination of the current selection in the window

ISelection adtSelection = window.getSelectionService().getSelection();

// 4) Determination of the active ABAP project

IProject project = ProjectUtil.getActiveAdtCoreProject(adtSelection, null, null, IAbapProject.ABAP_PROJECT_NATURE);

7.4.8.3 Determine the source code oft he active editor

// 1) Determine the active editor

IAdtFormEditor editor = (IAdtFormEditor)PlatformUI.getWorkbench()

.getActiveWorkbenchWindow().getActivePage()

.getActiveEditor();

// 2) Determine the editor's document

IDocument document = editor.getAdapter(AbapSourcePage.class).getDocument();



// 3) Determine the source code
String code = document.get();

7.4.8.4 Execute the transaction code

// 1) Determine the user setting of navigation to the Eclipse Editor for supported development objects

boolean navigateToEclipse =

```
com.sap.adt.sapgui.ui.internal.Activator.getDefault()
```

.getPreferenceStore()

.getBoolean(com.sap.adt.sapgui.ui.internal.PreferenceInitializ

er

.PREF_KEY_USE_ECLIPSE_NAVIGATION);

// 2) Execution of the transaction code

AdtSapGuiEditorUtilityFactory

.createSapGuiEditorUtility()

.openEditorAndStartTransaction(project,TRANSACTION_NAME, avigateToEclipse);

7.4.8.5 ABAP Scan Services – Check whether the token is a keyword

// 1) Determine the instance of AbapSourceUI
IAbapSourceUi sourceUi = AbapSourceUi.getInstance();

// 2) Determine the instance of SourceScannerServices

IAbapSourceScannerServices = sourceUi.getSourceScannerServices();

// 3) Determine the active ADT editor

editor = (IAdtFormEditor)PlatformUI.getWorkbench()

.getActiveWorkbenchWindow().getActivePage().getActiveEditor();

// 4) Locate the document from the editor

IDocument document = editor.getAdapter(AbapSourcePage.class).getDocument();

// 5) Check if the token at the offset position is a keyword (or not)



Boolean isKeyword = scannerServices.isKeyword(document,OFFSET);
// 6) Determine the next token based on the offset position
Token nextToken = scannerServices.getNextToken(document,OFFSET);
// 7) Check if the next token is a keyword (or not)
isKeyword = scannerServices.isKeyword(document,nextToken.offset);

#### 7.4.8.6 Identify the project and show the selection dialog

```
// 1) Determine the shell
Shell shell = PlatformUI.getWorkbench().getActiveWorkbenchWindow().getShell();
// 2) Show the selection dialog and determine the selected project
IProject chosenProject = AbapProjectSelectionDialog.open(shell, null);
```

#### 7.4.8.7 Identify the users and call up the selection dialog

// 1) Determine User Service

IAdtUserServiceUI adtUserService =
AdtUserServiceUIFactory.createAdtUserServiceUI();

// 2) Call up the user selection dialog and determine the selected users

```
String[] users = adtUserService.openUserNameSelectionDialog(null, false,
project,"");
```

#### 7.4.8.8 Logging on tot he ABAP System

// 1) Adapting an IProject Object to an IAbapProject Object

final IAbapProject abapProject = project.getAdapter(IAbapProject.class);

// 2) Check the login status with automatic login dialog if no one is already available

// Registration is available

IStatus logonStatus = AdtLogonServiceUIFactory.createLogonServiceUI()

.ensureLoggedOn(abapProject.getDestinationData(), PlatformUI.getWorkbench()

```
.getProgressService())
.isOK();
```

### 7.4.9 Project Set-up with Maven

In addition to the standard options for developing plug-ins, features and update sites that come with *the Eclipse IDE for RCP and RAP Developers*, there is also the option to use Eclipse Tycho. Eclipse Tycho is a collection of plug-ins for Apache Maven. Further information can be found on Tycho's project page. A tutorial on plug-in development with Tycho can be found here.



# **Authors**

Finally, we would like to introduce ourselves as a team of authors and continuous contact persons regarding any questions in the context of the ABAP development tools.

But first of all, we would like to take this opportunity to express our special thanks to **Marc Zimmek** (Claas KGaA), **Jens Zähringer** (Nagarro) and the DSAG office, who supported us in the final phases of the creation with their active proofreading.

In addition, we do not want to miss the opportunity to highlight the close, constructive and uncomplicated cooperation with colleagues at SAP. Many thanks to **Thomas Fiedler** (Product Owner ADT), **Wolfgang Wöhrle** (ADT documentation incl. configuration and installation), **Yannic Soethoff** and **Sebastian Ratz** (both ADT under Mac OS).

Name	Company and job title	Brief description of your activities and experiences with ADT
Michael Biber	R+V Allgemeine Versicherungen AG	First beginnings in 2016 in self-study. Gradually, increased use. Exclusive use today.
	System Developer and Application Manager	Since approx. 2018 support of oomph configurations for the SAP area.
Jörg Brandeis	Brandeis Consulting GmbH Managing Director, Trainer, Developer and Consultant	Jörg Brandeis is Managing Director of Brandeis Consulting GmbH in Mannheim, which offers training and consulting in the areas of SAP BW/4HANA, SAP HANA, Modern ABAP and SQLScript. He is the author of SQLScript for SAP HANA (SAP Press)



Uwe Fetzer	Silesia Gerhard Hanke GmbH & Co. KG Team Lead Global SAP Development & Analytics	ADT has been used privately since the beginning (co-author of the SAPLink extension for Eclipse), and has been using ADT exclusively with employers for five years
Thomas Foehn	Zürcher Kantonalbank	Responsible for ABAP development guidelines and best practices. No widespread use of ADT yet. Use depending on employee preference, even now an intensive user
Sebastian	msg systems ag	programming, designing, architecture;
Freilinger-Huber	Principal IT	Use of Eclipse (ADT) since its introduction;
	Consultant	Pushing the use of ADT within the company and divisions
	DSAG Speaker	
	Committee Development	Organizational management of the DSAG- ADT guide
Florian Henninger	FIS	programming, designing, architecture;
	Informationssyste me und Consulting GmbH	Use of Eclipse (ADT) since the beginning;
		Main area Odata/fiori elements
	Senior Consultant Development	
Armin Junge	msg systems ag	Architecture, Design, Implementation
	Lead IT Consultant	Intensive use of ADT for many years. Pushing the use in the area and at the customer's site.



Michael Keller	educatedBytes GmbH CEO, Consultant, Developer, Trainer	Michael Keller has been working for many years as a consultant, developer and trainer in the logistics environment of SAP ERP systems. Through development with the ABAP programming language, he became acquainted with the ABAP Development Tools for Eclipse in 2017 and has been working with this development environment ever since.
Peter Luz	Robert Bosch GmbH Development Engineer and - architect	ABAP developer and coordinator for centrally developed components and solutions in the area of SAP Logistics Execution. Fan of ABAP-OO and ABAP Unit Tests. Entry into ADT 2015. With ADT, the creation of high-quality and modern code has become much more efficient and faster.
Dominik Panzer	Intense AG Senior Manager/ Head of Development/ Agile Technical Coach	Long-time ADT users and often buried deep in legacy codebases to transform them and bring them into the new cloud world. Therefore, he really appreciates the refactoring features of ADT. In addition to technical topics, clean code and software architecture, team-oriented topics such as agile games, coding dojos, pair and mob programming are also close to his heart and passes them on as a coach.
Lukasz Pegiel	Hager Group Digital & Information Manager Poland	<ul> <li>I'm responsible among others for:</li> <li>local teams management, which includes ABAP Development Team, Net Development Team, IT support and BIM (Building Information Modeling Team)</li> <li>Collaboration with External Resources Providers</li> <li>S/4HANA custom code adaptation.</li> </ul>



Dr. Wolfgang Röckelein	Axians NEO Solutions & Technology	ADT are used by me and my colleagues in product development and in customer projects for our products across the board.
	Senior Architect Product Development	
Björn Schulz	REWE digital GmbH	Product owner and developer for ABAP, Steampunk and BTP. Only ADT for almost four years and always striving to motivate colleagues in the direction of ADT and to
	Product Owner (SAP Development & Technology)	"sell" the many advantages.
Ludwig Stockbauer-	msg systems ag	Architecture, Design, Implementation
Wum		ADT has been in use for many years
	Senior IT Consultant	Passionate add-on developer for ADT
Bärbel Winkler	Alfred Kärcher SE & Co. KG	ABAP Development & Guidelines. Not yet a widespread use of ADT and even only "rudimentary" experience, but I am using Eclipse more and more often and would like
	Consultant SAP Development	to motivate others to do the same.

Table 6 Authors



## **Imprint**

We expressly point out that this document cannot anticipate and cover every regulatory requirement of all DSAG members in all business scenarios. In this respect, the topics and suggestions addressed must naturally remain incomplete. The DSAG and the authors involved cannot assume any responsibility for the completeness and suitability of the suggestions for success.

This publication is protected by copyright.

Unless expressly stated otherwise, all rights are vested in:

Deutschsprachige SAP® Anwendergruppe e.V. Altrottstraße 34 a 69190 Walldorf | Deutschland Telefon +49 6227 35809-58 Telefax +49 6227 35809-59 E-Mail info@dsag.de dsag.de

Any unauthorized use is not permitted. This applies in particular to duplication, editing, distribution, translation or use in electronic systems/digital media.

© Copyright 2023 DSAG e.V.

