Available online www.jsaer.com

***Journal of Scientific and Engineering Research*, 2022, 9(4):184-191**

Research Article

ISSN: 2394-2630
CODEN(USA): JSERBR

S/4HANA Migration Introduction, Strategy, and Planning– Part 3

Deepak Kumar

Wilmington, USA
Email: Deepak3830@gmail.com

Abstract SAP S/4HANA is SAP's next-generation ERP suite designed to provide businesses with enhanced speed, agility, and innovation capabilities. Data migration to S/4HANA is a critical transition component, ensuring that existing data from legacy systems is seamlessly transferred to the new platform. SAP S/4HANA's in-memory computing capabilities enable businesses to access real-time insights into their operations, facilitating expedited decision-making and responsiveness to market changes. Real-time analytics empower organizations to recognize trends, anticipate customer requirements, and optimize processes to achieve superior outcomes. The SAP S/4HANA Migration Cockpit is part of SAP S/4HANA and SAP S/4HANA Cloud and its use is included in these licenses. It is a ready-to-use solution that contains a comprehensive set of preconfigured migration objects such as customer, supplier, material, cost center, and so on. The proven methodology is integrated into the SAP Activate roadmap Transition to SAP S/4HANA. It is part of the SAP Model Company service and represents best practices in data transfer.

Keywords SAP, SAP S/4 HANA, S/4HANA Migration Cockpit

1. Introduction

S/4HANA Migration Cockpit – The S/4HANA Migration Cockpit is a comprehensive tool provided by SAP to facilitate the migration process from legacy SAP ERP systems to SAP S/4HANA. It offers a user-friendly interface and a set of preconfigured templates and tools to simplify and streamline the migration process. The Migration Cockpit comes pre-installed with SAP S/4HANA and does not require additional installation or setup. It contains a comprehensive set of preconfigured migration objects such as customer, vendor, material, and financial data, which can be easily mapped and migrated to the new system. It includes built-in data validation checks and error-handling capabilities to ensure data accuracy and integrity throughout the migration process.

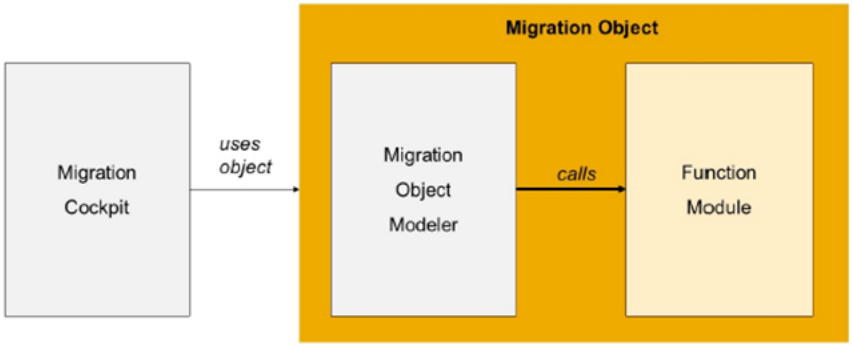
ETL – ETL stands for Extract, Transform, Load, which represents a common process in data warehousing, migration, and analytics. The ETL process plays a critical role in data integration, consolidation, and analysis, enabling organizations to derive insights, make informed decisions, and drive business outcomes based on their data assets. ETL tools and platforms automate and streamline these processes, providing features for data extraction, transformation, loading, scheduling, monitoring, and error handling.

RFC - A Remote Function Call (RFC) is a communication protocol used in SAP systems to enable communication between different systems or between different components within the same system. RFC allows programs written in different languages or running on different platforms to communicate with each other seamlessly.

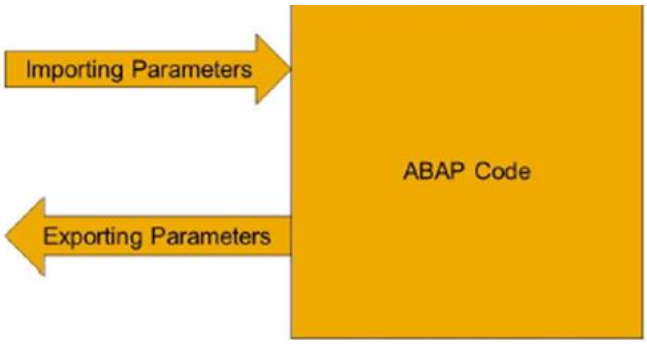
Pre-requisite: To understand this paper thoroughly prerequisite is– S/4HANA Migration Introduction, Strategy, and Planning– PART 2 and PART 3.

2. Migrating user objects using a custom Functional Module

In addition to utilizing the Migration Object Modeler for the adaptation of existing migration objects, it also facilitates the creation of custom migration objects. For instance, migration objects are generated when there is a necessity to migrate extensive custom applications or supplement standard functionalities of SAP S/4HANA that are not included by pre-existing migration objects.



Function modules are reusable procedures crafted using ABAP within the ABAP Workbench. They feature importing parameters to receive data and exporting parameters to communicate results effectively. When employing a function module within a migration object, the importing parameters relay the data of the data record being migrated. Additionally, the function module should provide a simulation mode and an importing parameter to indicate whether it is being called in simulation mode. Furthermore, an exporting parameter is utilized to convey any error messages generated by the function module. The ABAP code is responsible for validating the data's integrity before inserting it into the database table.



When you utilize a function module in a migration object, the importing parameters carry the data of the data record being migrated. It's like passing along the details you want to move. Additionally, the function module needs to have a simulation mode and an importing parameter to indicate if it's operating in simulation mode. This helps test without actually making changes. Any error messages triggered by the function module are stored in an exporting parameter. Think of it as a way to catch issues that pop up. The ABAP code is responsible for ensuring the data is valid before adding it to the database table, kind of like double-checking everything before it gets permanently stored.

Function modules utilized in migration objects are required to have an exporting parameter with the type BAPIRET2. This parameter serves as a conduit for the function module to relay a message, which can either be a success message or an error message. These messages are crucial for conveying the outcome of the function module's execution. The migration cockpit processes these messages and incorporates them into the migration log, providing a comprehensive record of the migration process and any associated messages or errors encountered. Instead of utilizing exceptions, which are prohibited, this parameter serves as a substitute for conveying messages from the function module.

The function module is responsible for guaranteeing the consistency of data prior to its insertion into the database. For instance, the provided source code extract, labeled "The Source Code," showcases a verification process to ensure that the intended flight connection does not already exist. However, this is merely one of the

numerous checks that even this basic function module must undertake. Additional examples include verifying the existence of departure and arrival cities and airports, as well as confirming that the departure and arrival airports are distinct from each other.

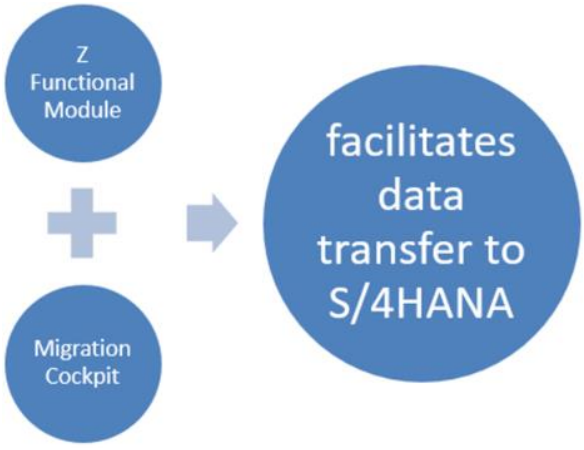
If the function module encounters an error, it is imperative to report it using the return structure and a corresponding message. Each component within the return structure serves a distinct purpose:

- Type (TYPE): Specifies the message type, such as "S" for success or "E" for error.
- ID (ID): Identifies the message ID, which helps categorize and classify the message.
- Number (NUMBER): Indicates the message number, providing further specificity within the message ID.
- Message (MESSAGE): Contains the actual message text, conveying relevant information about the encountered issue.
- Log Number (LOG_NO): Associates the message with a specific log entry, facilitating traceability.
- Log Message Number (LOG_MSG_NO): Provides a message number for the log entry, aiding in retrieval and reference.
- Message Variable 1 (MESSAGE_V1): Placeholder for variable data within the message text.
- Message Variable 2 (MESSAGE_V2): Another placeholder for variable data within the message text.
- Message Variable 3 (MESSAGE_V3): Additional placeholder for variable data within the message text.
- Message Variable 4 (MESSAGE_V4): Yet another placeholder for variable data within the message text.

By utilizing these components effectively, the function module can communicate error messages comprehensively, enabling the identification and resolution of encountered issues. Messages are identified by a combination of a message class and a number. A message class encompasses all the message texts needed for a specific application. Within the class, each message is assigned a unique number. Messages can incorporate placeholders, which are denoted by &1 to &4. These placeholders allow for the dynamic insertion of variable data into the message text, enhancing its flexibility and adaptability.

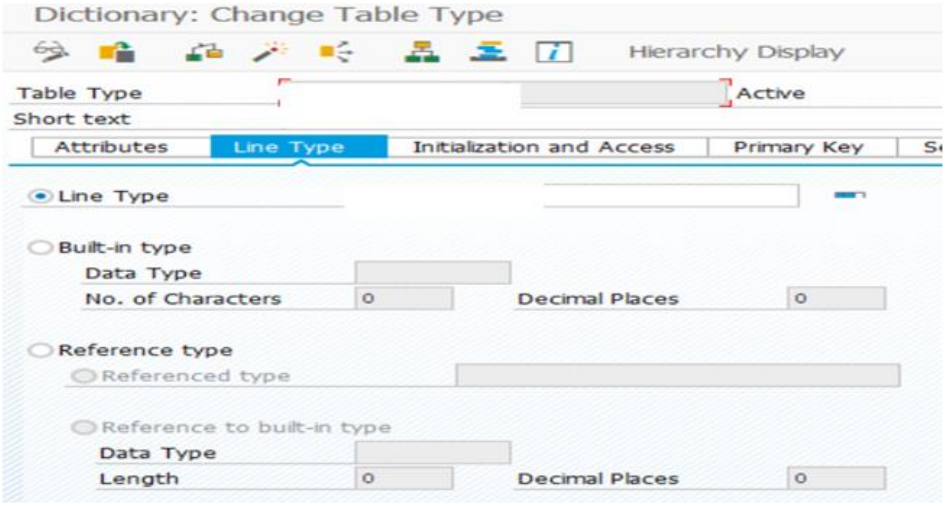
In the absence of errors, the function module proceeds to insert the data into the relevant database table or tables. However, it refrains from executing a COMMIT WORK statement. Instead, the responsibility of executing this statement lies with the Migration Cockpit. This approach ensures that data integrity is maintained throughout the migration process and that database transactions are managed appropriately by the central migration tool.

3. Create the migration object



To create your own migration object, you must first create a project using the Migrate Your Data app. Use the staging table approach to supply the data. When you create the project, you switch to transaction LTMOM to create the migration object.

Run SE11 t-code. Chose Data type option and Table type. Set Line Type as your database table. Save and activate.

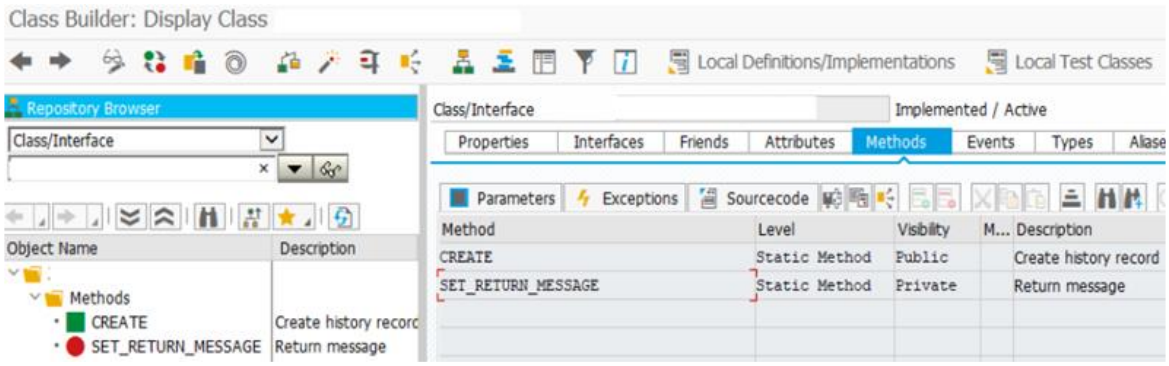


Create Class: Run SE80 t-code, create the custom class

Method	Level	Visibility	Description
CREATE	Static Method	Public	Meaningful Description
SET_RETURN_MESSAGE	Static Method	Private	Add return message

Parameter	Type	Typing Method	Associated Type
CREATE			
IT_ZSDVARNAme	Importing	Type	ZSD_VARNAM_TT
ET_MESSAGES	Exporting	Type	BAPIRET2_T
SET_RETURN_MESSAGE			
IV_TYPE	Importing	Type	BAPI_MTYPE
IV_ID	Importing	Type	SYMSGID
IV_NUMBER	Importing	Type	SYMSGNO
IV_MESSAGE_V1	Importing	Type	SYMSGV
IV_MESSAGE_V2	Importing	Type	SYMSGV
IV_MESSAGE_V3	Importing	Type	SYMSGV
IV_MESSAGE_V4	Importing	Type	SYMSGV
ET_MESSAGES	Exporting	Type	BAPIRET2_T

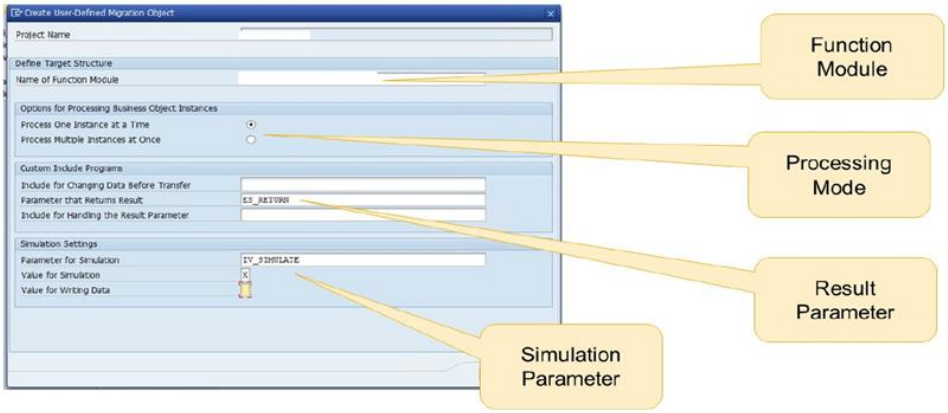
Symbol	Text
001	Populate mandatory data
002	Record was created



Create Function Group, and FunctionModue, and set export/import parameters

Import			
Parameter Name	**Typing**	**Associated Type**	**Short Text**
IV_TEST_RUN	TYPE	BAPIE1GLOBAL_DATA-TESTRUN	Test run
IT_VARNAME	TYPE	ZSD_HISTCOND_TT	Name
		Export	
ET_MESSAGES	TYPE	BAPIRET2_T	Return messages

To create a new migration object, switch to change mode in your project and choose Project → Create Migration Object → From Template → Migrate Data Using Files/Staging Tables. Enter a name for the new migration object. The object name always has the project's mass transfer ID as a suffix. You must also enter a description for the object. The description of the object is important. It is the text that appears as the name of the migration object in the Migration Cockpit.



When generating your migration object, it's imperative to furnish the function module's name along with supplementary details.

In Processing Mode, you have the option to determine whether data records are transmitted to the function module individually or in batches. Typically, batch processing yields faster results compared to handling individual objects. However, it's essential to note that the function module must be compatible with batch processing by accommodating multiple records simultaneously. Regrettably, the function module depicted in the Object Details section does not currently support this feature.

Result Parameter refers to the name of the parameter within the function module that holds messages. This information is crucial for the Migration Object Modeler as it utilizes it to manage and process messages effectively.

The simulation Parameter refers to the parameter within the Migration Cockpit that dictates whether the function module operates in simulation mode or executes the actual data migration process. It is essential to specify both the parameter name and its corresponding value for the simulation mode. This enables the Migration Cockpit to appropriately simulate the migration process before performing the actual data migration.

Key Field: Indicates whether the field serves as a key field. Key fields, when combined, uniquely identify each record, and they must be positioned at the start of the structure.

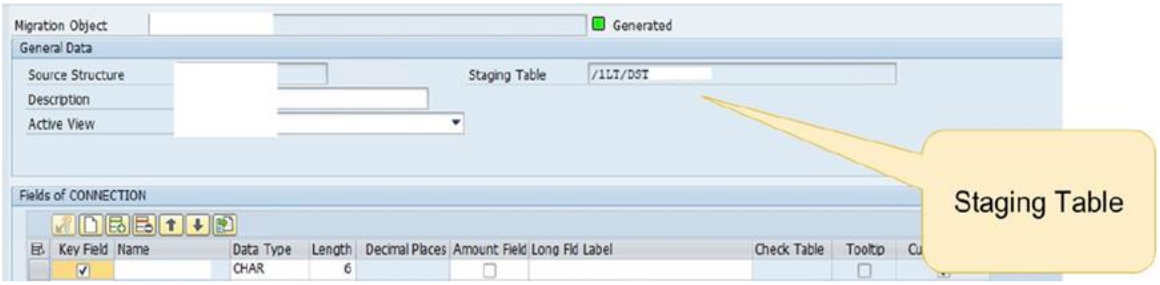
Field Name: Every field must possess a distinct identifier.

Data Type: Each field necessitates a defined data type. Certain types, like CHAR and NUMC, necessitate specifying a length, while others, like INT4 and DATS, come with predefined lengths. For DEC data type, both length and the number of decimal places are required.

Column Header: This designation represents the column heading showcased in the file template. When utilizing staging tables, it manifests as the field description within the table definition in the SAP HANA Studio.

Group Text: This designation denotes the field group displayed within the file template.

By default, only key fields within the source structure are designated as mandatory. To enforce additional fields as mandatory, navigate to the source structure view and modify the field attributes accordingly.



When generating a source structure within a staging table project, the system automatically generates the corresponding staging table in the database. This process utilizes the database connection configured in the migration project settings. To facilitate later testing of the object, it's essential to populate the staging table. Crucially, note that the staging table is shared between both the migration project and the Migration Object Modeler.

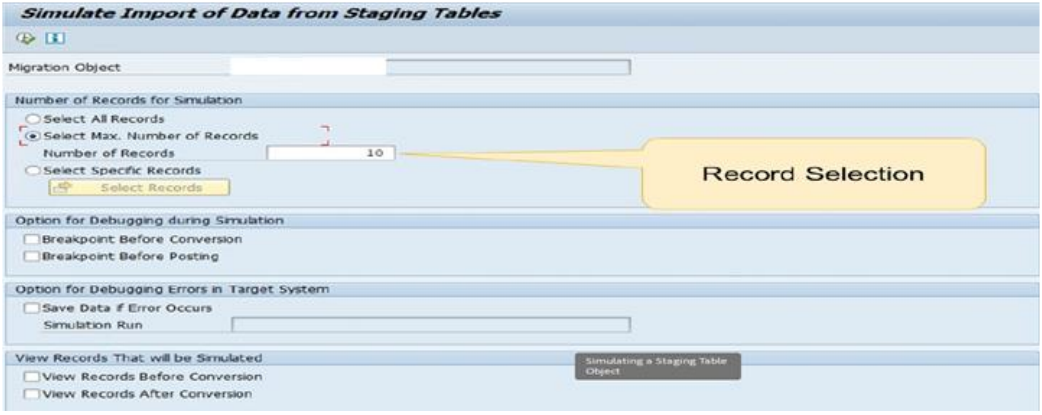
The target structure is delineated by the parameters of the function module. Within the Structure Mapping segment of the migration object, drag the Connection node onto both the Header and the R_IS_CONNECTION node. Mapping fields typically involve a straightforward move operation for most attributes. However, it's noteworthy that the flight number is depicted differently in the source and target structures.

To evaluate the migration object, download the file template using the icon displayed in the application toolbar. The columns within the file template align with the fields present in the source structure.

You can fill the template with sample data and upload it using the Simulate function. You can then simulate the migration.

To document your migration object, choose Goto → Show/Edit Documentation. The documentation is then available to users in the Migration Cockpit.

During the testing phase of your object, it's crucial to select the file template containing the test data. Utilize standard debugging functions for evaluation purposes. If you wish to debug the function module responsible for data posting, activate the "Breakpoint Before Posting" option. This feature triggers the debugger a few lines prior to the actual function call. To verify the accuracy of your rules and conversions, employ the "View Records After Conversion" function. When running the simulation, the system presents a dialog box showcasing the converted values for examination.



When simulating the migration of a staging table object, many settings remain consistent with simulating a file-based object. However, the data selection process differs. With the staging table approach, you have the flexibility to choose from various data selection options. These options include selecting all objects, specifying particular records by entering specific key values, or opting to migrate a certain number of objects at random.

The staging table serves as a shared resource accessible to both the simulation tool in the Migration Object Modeler and the migration project within the Migration Cockpit. Consequently, it's imperative to prevent conflicts between your testing activities and any actions undertaken in the Migration Cockpit.

4. Conclusion

In conclusion, the migration to S/4HANA represents a pivotal step for enterprises seeking to modernize their ERP landscape. The process involves intricate steps and considerations, ranging from the selection of migration approaches to the actual transfer of data. Migrating data to S/4HANA using direct transfer offers a streamlined method for organizations to transition their critical information. By leveraging direct transfer capabilities, businesses can minimize complexities associated with staging tables and expedite the migration process. The steps for migrating data to S/4HANA via direct transfer encompass configuring the migration project, selecting relevant objects, and executing the migration process. Furthermore, transferring projects across systems is essential for maintaining consistency and continuity throughout the migration journey. By adhering to proper transport protocols and release procedures, organizations can effectively move their projects from development to production environments.

Declarations

Ethics approval and consent to participate: Not Applicable

Consent for publication: All authors have consent to submit this paper to the Journal of Cloud Computing. Also, we confirm that this paper or any part of this paper was not submitted anywhere.

Availability of data and materials: Not Applicable

Competing interests: Not Applicable

Funding: Not Applicable

References

[1]. IlianaOlvera7, "Part 1: SAP S/4HANA migration cockpit – Migrating data using staging tables and methods for populating the staging tables (transaction LTMC – deprecated with SAP S/4HANA 2021)," SAP Community, Dec. 02, 2019. https://community.sap.com/t5/enterprise-resource-planning-blogs-by-sap/part-1-sap-s-4hana-migration-cockpit-migrating-data-using-staging-tables/ba-p/13415612

[2]. i003814, "Part 2: SAP S/4HANA migration cockpit - Using SAP Data Services to load data to the staging tables," SAP Community, Nov. 29, 2019. https://community.sap.com/t5/enterprise-resource-planning-blogs-by-sap/part-2-sap-s-4hana-migration-cockpit-using-sap-data-services-to-load-data/ba-p/13458983

[3]. "Data Migration in SAP S4 HANA | PDF | Data Quality | Information Technology," Scribd. https://www.scribd.com/document/443298739/Data-Migration-in-SAP-S4-HANA

[4]. "SAP S/4HANA Migration Guide: Steps you Can Take Now to Prepare Migration Guide." https://us.nttdata-solutions.com/hubfs/IB-NN-S4HANA-Migrations/MigrationGuide-itelligence-SAP-S4HANA.pdf

[5]. IlianaOlvera7, "Part 1: Migrate your Data – Migration Cockpit (SAP S/4HANA 2020 and higher and SAP S/4HANA Cloud, public edition), Migrate data using staging tables and methods for populating the staging tables with data," SAP Community, Mar. 10, 2021. https://community.sap.com/t5/enterprise-resource-planning-blogs-by-sap/part-1-migrate-your-data-migration-cockpit-sap-s-4hana-2020-and-higher-and/ba-p/13501516

[6]. AleSabidussi, "Part 3: SAP S/4HANA Migration Cockpit - Using SAP HANA Smart Data Integration (SDI) to load data to the staging tables," SAP Community, Dec. 02, 2019. https://community.sap.com/t5/enterprise-resource-planning-blogs-by-sap/part-3-sap-s-4hana-migration-cockpit-using-sap-hana-smart-data-integration/ba-p/13454249

[7]. alexey_danshin, "Migrating user object using Your own Functional Module in S/4HANA Migration Cockpit," SAP Community, Apr. 23, 2020. https://community.sap.com/t5/enterprise-resource-planning-blogs-by-members/migrating-user-object-using-your-own-functional-module-in-s-4hana-migration/ba-p/13451459

[8]. "SAP S/4HANA Migration Cockpit - Creating Your Own Function Modules in LTMOM," SAP. https://www.sap.com/documents/2020/05/44a27c28-977d-0010-87a3-c30de2ffd8ff.html

[9]. JK, "Re: Discover the S/4HANA Migration Cockpit Migration Object Modeler (OnPremise) /NLTMOM," SAP Community, Mar. 06, 2018. https://community.sap.com/t5/enterprise-resource-

planning-blogs-by-members/discover-the-s-4hana-migration-cockpit-migration-object-modeler-onpremise/bc-p/13320033

[10]. ThFiedler, "ABAP custom code adaptation for SAP HANA – The efficient way," SAP Community, Apr. 22, 2016. https://community.sap.com/t5/enterprise-resource-planning-blogs-by-sap/abap-custom-code-adaptation-for-sap-hana-the-efficient-way/ba-p/13195113

[11]. OlgaDolinskaja, "Custom code analysis for SAP S/4HANA with SAP Fiori App Custom Code Migration," SAP Community, Feb. 27, 2019. https://community.sap.com/t5/enterprise-resource-planning-blogs-by-sap/custom-code-analysis-for-sap-s-4hana-with-sap-fiori-app-custom-code/ba-p/13402751

[12]. mahesh_sardesai, "New Installation of SAP S/4HANA 2020FPS0 – Part 5 – Direct Data Transfer using Migration Cockpit," SAP Community, Oct. 31, 2020. https://community.sap.com/t5/enterprise-resource-planning-blogs-by-sap/new-installation-of-sap-s-4hana-2020fps0-part-5-direct-data-transfer-using/ba-p/13484024