Authors:
Kiran Kumar Khamitkar
Praveen P

Software Design Description

Consultant Profile Integration

Version	Status	Date	Reviewed/Approved at / by
0.1	In draft	12/02/2020	Praveen and Kiran
0.2	Update availability column details	07/05/2020	Praveen and Kiran
0.3	Update risk and assumptions	12/08/2020	Praveen and Kiran
0.4	Update availability replication model	19/11/2020	Praveen and Kiran
0.5	Update frameworks and used components	22/09/2021	Kiran

INTERNAL / CONFIDENTIAL

Storage location for temporary versions of this architecture concept document:

Final version:

Read before using this template!

How to use this template

The purpose of a design doc is

(a) develop design concepts for the target component,
(b) enable review of the key concepts,
(c) support alignment between stakeholders,
(d) record trade-offs and reasons for major design decisions as needed.
(e) In addition we use this document as a 'Guide to the code' when the implementation is done.

Consider these guidelines when using this template:

- Document only what is needed by reviewers and required for development of high quality software
- Less is better! Write short / concise and so that it is easy to understand. Focus on key design challenges, not completeness (large documents are a waste and no one will read them!)
- Write for ease of understanding but don't waste time on unnecessary polishing.
- 'Pair Design' recommended (= develop key concepts together with a colleague)
- Document design details & 'guide to the code' after the implementation. Only write what is not contained in the code/system already
- A template is a template and not a form. Only fill what is needed and relevant.
- This template is a base template. We expect large development units to derive their own versions that are specific to their environment / technology.

Some guidelines on reviews

- An Expert Review is mandatory only for chapter 2
- Differentiate between **expert review** (with *few* experts, deep review, issues & changes likely) and **light review / information rollout** (many people, no/few issues expected).
- Expert Review Participants: only experts & *directly affected people* (provider and consumers)
  - Design for team topic: review with dev team + stakeholders
  - (Public) Interfaces / Cross-alignment: Providers and consumers
  - Do a real meeting (not just mail / doc) to discuss the key design topics
  - Try for no more than 7 people!
- Light Review Participants: There can be many participants. Can be mail / doc based, meetings to discuss remaining issues are optional

Hidden Text contains important useful hints on how to fill in the template. Activate / de-activate these texts by pressing the following button in MS Word toolbar:

Contents

1 General Information 5
1.1 Stakeholders and Roles 5
1.2 References 5
1.3 IP Compliance and Patents 5
2 Design 6
2.1 Key Requirements and Design Goals 6
2.2 Context 6
2.3 Major Building Blocks 7
2.4 Design Challenges resulting from Non-Functional Requirements 11
2.5 User Interface 11
2.6 Used Components and Frameworks 11
2.7 Package/Development Component Concept 12
2.8 Upgrade / Migration / Compatibility 12
2.9 TCO Considerations 12
2.10 Compliance to Standards and Guidelines 12
3 Design Details Documentation 14
3.1 Implementation Strategy 14
3.2 Data Models 14
3.3 Integration Strategy 16
3.4 Implementation Details 21
3.5 Other Topics 32
3.6 Testability and Test Environment 41
3.7 Limitations 41
3.8 Other External Forces / Constraints and Assumptions 42
3.9 Open Points 43
3.10 Complex Algorithms and Applied Patterns 45
3.11 Design Alternatives and Trade-Offs 45
3.12 Guide to the Implementation 45
4 Appendix 45
4.1 Glossary 45
4.2 Customizing 46
4.3 Supportability Considerations 46
4.4 Error Analysis 46
4.5 Other 47

1 General Information

1.1 Stakeholders and Roles

Role	Name
Author(s)	Kiran Kumar Khamitkar, Praveen P
Architect	Praveen P
Product Owner	Raymond Amalraj
Information Developer	Asha Roberts
Quality Responsible (QE/QPE)	Tanuja Yaganti
IMS Representative	NA

1.2 References

Document Title	Date	Link	Comments
ACD Document	05.03.2020	Architecture Concept Document - RD 2003 - Version 1.0	
SDD Document on Consultant Profile	03.03.2020	Consultant Profile SDD	

1.3 IP Compliance and Patents

You must always store architecture and design documents 'IP safe' (currently cPro) so that (if needed) SAP can prove that a certain idea / concept was invented / designed at a certain point in time.

In addition, consider patents: Does your design extend the current state-of-the-art in any way? If you think so, or even if you think it might, please go to the 'Patents@SAP' Wiki and follow the process described there. Not only is it required of all employees to notify SAP if they make an invention, but participation in the SAP patent program is rewarded in several ways (including money) – so by taking part in the SAP patent program, you can help SAP protect its innovation and receive recognition along the way.

2 Design

This document is intended for developers to understand the HR source integration with target system Resource management (RM) Consultant Profile. This design assumes that SFSF (SuccessFactors) is the source system for HR data. Its also assumed that SFSF already has an integration with oneMDS for the employee basic data and oneMDS is the source for such data for all subscribers one of them being RM. For data involving availability the information would be available in oneMDS at a future point in time which can then be used to compute availability in the consuming system. Until then the Availability details will be upload using CSV file, would be the way to integrate in RM to get employee availability information.

2.1 Key Requirements and Design Goals

The Key requirements is to provide a way for customers to bring in below information from HR Source System to RM solution in SCP.

1. Employee Personal Data (Name, Email, Phone, Mobile)
2. Employee Org Data (Manager name, Cost Center, Org Unit)
3. Employee Job Details
4. Time dependency of information replicated has to be taken into account.
5. Availability Information

For points 1 to 4, the information would come in from workforce person data in oneMDS and for point 5 regarding availability the information would be uploaded in CSV file using separate application to maintain availability details.

For furthermore requirement details can be found below:
https://sapjira.wdf.sap.corp/browse/MODULAREPPMVALIANT-118

2.2 Context

The Consultant profile is one of the main entities that will be used by Resource manager to staff candidates within projects. In the consultant profile the employee master data would be enriched with other details that are very specific to Resource Management. For example, skills of the employee, role of the employee, internal and external work experience of the employee etc. The master data information would be sourced from the HR system through integration with Master data Service (oneMDS).
The Availability information would be uploaded in CSV file using separate application to maintain availability details.

2.3 Major Building Blocks

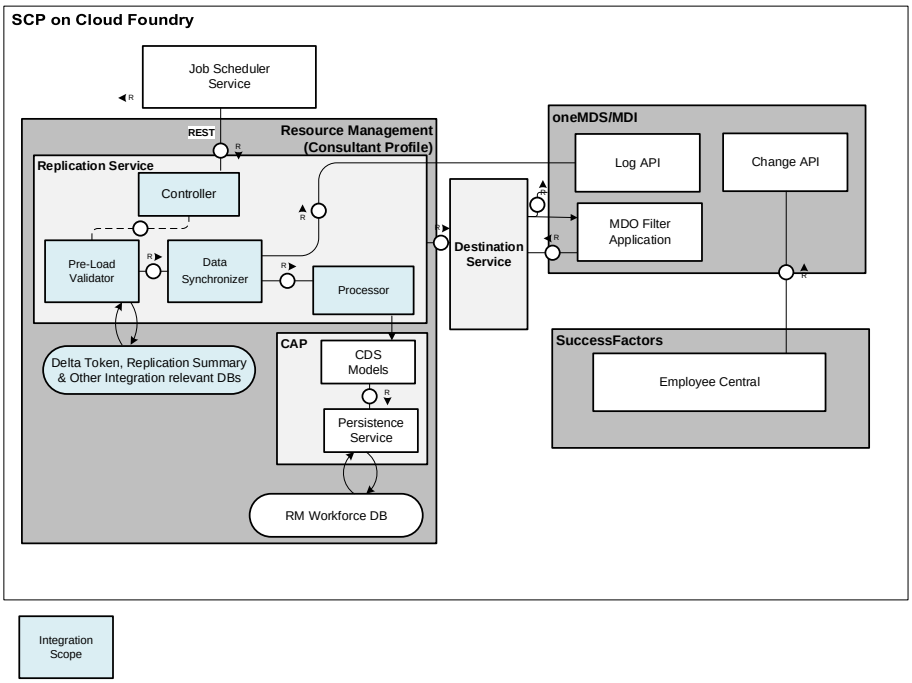


Fig1: Consultant Profile Integration Block Diagram

2.3.1 JOB Scheduler service

SAP Cloud Platform (SCP) offers Job Scheduler as a service, for managing your jobs that run either only once, or on recurring schedules.
A job is a collection of schedules with an action endpoint. A schedule is a one-time or a recurring entity within a job without an action endpoint. A service plan within Job Scheduler takes into account the number of schedules within a job.
Job Scheduler service, which you can execute using any runtime, schedules any REST endpoint actions or cloud foundry tasks in your application, helping leverage the potential of your application. As a developer, you can allocate more time to building business logic and workflows that resolve core business needs, while offloading one-time or long-running repeated tasks or routines to Job Scheduler.
We can use SAP Cloud Platform Job Scheduler to define and run job schedules for the application with endpoints hosted on the Cloud Foundry SAP Cloud Platform.

The Job Scheduler service also provides a dashboard where we can manage and monitor the jobs defined.

Multitenancy in Job Scheduler

Multi-tenancy in Job Scheduler enables Multi-tenant Applications in SAP Cloud Platform to register Jobs on behalf of tenants (or Sub Accounts) who subscribes to them. It also enables the multi-tenant application to infer the information (Tenant ID, SubDomain etc.) of the tenant (or Sub Account) for whom the Job has to be executed when Job Scheduler invokes its HTTP endpoint.

2.3.2 Consultant Profile in Resource Management (RM)

The resource management solution is a cloud-based solution built on Cloud Foundry platform in SAP Cloud Platform (SCP). It would have the following components from the purview of the consultant profile domain.

Replication Service

This is a PULL based replication to replicate the data from source system. The job scheduler would call a REST endpoint. The controller will get the request, to start the replication and invoke the pre-load validator. The pre-load validator is an asynchronous service which will read the configuration entries to determine the cost centers in RM for which the replication has to happen. The data synchronizer is responsible for connecting to oneMDS and uses LOG Api provided by service to replicate employee data. This data then undergoes a transformation in the processor component before mapping it to the CDS models and invoking the CAP persistence service to save the data in RM.

CAP

The “SAP Cloud Application Programming Model” is a framework of languages, libraries, and tools for building enterprise-grade services and applications on SCP. It guides developers through proven best practices and a great wealth of out-of-the-box solutions for recurring tasks.

CDS is a family of domain-specific languages (CDL, CQL) and corresponding notations (CSN, CQN, CXN). It serves as the modelling language, for example, for capturing domain models and service definitions in a conceptual way, and hence as the very backbone of CAP.

The Persistence Service provides an API to access the data stored in the database, independently of the concrete database type in use. It exposes the data via the same API offered by CDS services, which is based on CDS QL statements. This consistent approach allows the Generic Providers of a CDS service, to simply forward the statements they received to the persistence service, if the data is stored in the database. Eventually, the persistence service executes the required operations on the database leveraging the CDS Data Store.

2.3.3 oneMDS/MDI

Currently, the integration of master data between SAP applications is done as a point-to-point integration on customer site. Therefore, for every customer implementation, the master data integration needs to be set up from scratch.
This Technology Guideline(TG) TG22 targets to provide the ONEmasterdata service(oneMDS) as central access point ensuring the harmonization of master data throughout all involved SAP applications. This will ensure that customers need to maintain any master data changes only once.

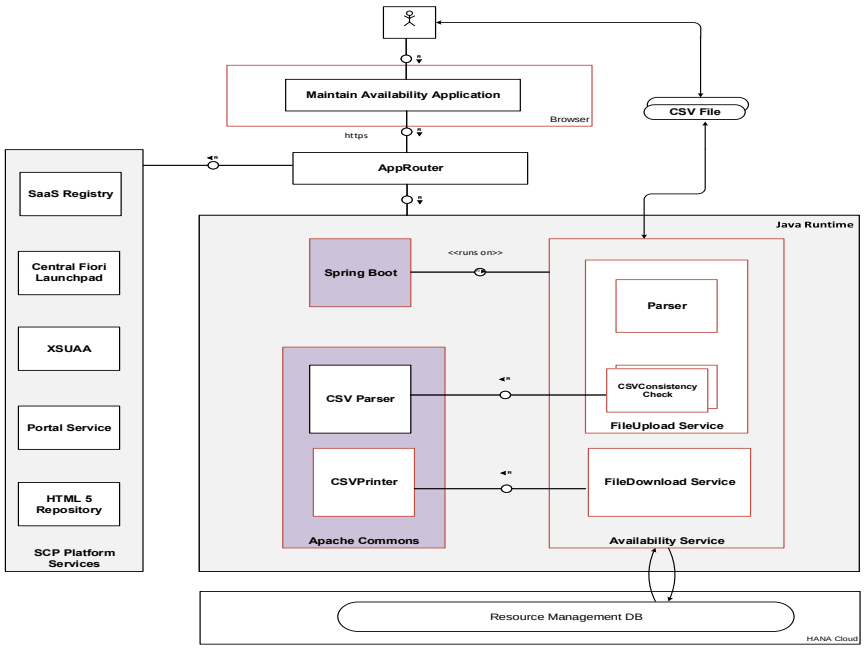
The most important APIs used to integrate applications with MDI are the following:

- The Change API is used for issuing change requests-> in the case of our solution SFSF EC would push data into oneMDS via the Change API.
- The Log API is used for reading recent change events to the data -> in the case of RM solution, there would a periodic pull of employee data from oneMDS via the Log API.

Integration with *oneMDS* will basically have below functionalities:

1. Load initial data
2. Delta load; for any changes in source

2.4 Major Building Blocks - Availability File Upload & Download



2.4.1 Spring Boot

Framework to build Java Applications.
More details are available at following link.

2.4.2 Approuter

Central component that manages / routes all HTTP requests.

2.4.3 Portal Service

Required by FLP e.g. for personalization.

2.4.4 XSUAA

User Account and Authentication Service (shared component of SCP).

2.4.5 Fiori Launchpad

Used as entry point to display all apps delivered with Resource Management

2.4.6 HTML5 Repository

Used to store / publish the UIs.

2.4.7 Apache Commons CSVParser

Used to parse the CSV files to import and export availability csv.
More details are available at following link.

2.4.8 Maintain Availability Application

The application to *maintain* availability *manually* (including labels) is implemented in Fiori Elements.

Availability Upload App

The application that triggers the CSV file import is implemented as a freestyle SAP UI5 application.The AvailabilityFileUpload app is based on the *FileUploader* control provided by UI5. The app uses the ajax-based control to be able to set the following header:

1. CSRF Token

Availability Download App

The application that enables the user to download the availability summaries that are available in the database has been implemented as a freestyle SAP UI5 application.

2.4.9 File Upload Service

The *File Upload Service* is based on Spring Boot and implements a REST controller that is able to accept a CSV file that has the following columns:

Name of Column in CSV	Usage
resourceId	Mandatory
workForcePersonExternalId	ignored
firstName	ignored
lastName	ignored
s4costCenterId	Mandatory
workAssignmentExternalId	Mandatory
startDate	Mandatory
plannedWorkingHours	Mandatory
nonWorkingHours	Mandatory

2.4.10 File Download Service

Whenever a download of the availability data is requested, the user has to either provide the cost enter or workforcepersonId as mandatory input along with the time period to retrieve dependent attributes of the availability. Download template will of the format above defined.

2.5 Design Challenges resulting from Non-Functional Requirements

High Availability of services like oneMDS and backend services like HANA DB, Audit Log etc is required.

2.6 User Interface

To be discussed and aligned by PO/UX if there needs to be a new application for error monitoring and error processing

2.7 Used Components and Frameworks

To be updated

Name	Description
MDI	Master Data Integration used for integration of master data with RM via a source system. In this case *WorkforcePerson* (Employee Data) is coming from SuccessFactors as the source and *costcenter* is coming from S/4 HANA as the source system.

SAP CAP NG Framework	Framework to create SAP Cloud applications based on Java or NodeJS. Features used: CDS, CDS4J, Spring Boot, Java, Generic OData Provider
SAP Connectivity Service	SAP Cloud Platform Connectivity provides services for the Cloud Foundry environment, the Connectivity service and the Destination service.
SAP Cloud Platform Java Runtime	Java Runtime
SAP HANA with HDI	Managed HANA DB
Fiori Elements v4	Used to create frontends based on annotations provided on the exposed CDS entities
Spring Boot	Framework to build Java Applications
XSUAA	User Account and Authentication Service (shared component of SCP)
Cloud logging	The Cloud Logging service offers managed logging service instances for shipping logs from Kubernetes, Cloud Foundry, or any other source.
Job Scheduler Service	Enables periodic job execution in the context of PaaS tenant or SaaS tenants.
Destination Service	Enables connectivity of internal / external services/destinations from within applications
Audit logging service	Enables logging auf audit-relevant events
SaaS Registry Service(LPS)	BTP Kernel Service SaaS Registry Integration for Commercialization of the SaaS solution / PaaS service. Enables on-boarding of SaaS tenants of end customers. Enables sales of the SaaS solution and subscription by end customers

2.8 Package/Development Component Concept

To be updated

2.9 Upgrade / Migration / Compatibility

Not relevant

2.10 TCO Considerations

Not relevant

2.11 Compliance to Standards and Guidelines

Not relevant

2.11.1 Applied Architecture / Design Guidelines

Name	Version/Date	Link

Naming Convention (for CDS, HANA, Java and others)Architecture Guideline of ….	Oct. 30th, 2019	link

2.11.2 Approved deviations

3 Design Details Documentation

3.1 Implementation Strategy

- Use SAP CP Cloud Foundry as Runtime
- Use SAP Re-usable oneMDS service
- Use HANA as a Service (HaaS) as the database with multitenancy support offered by Managed HANA Service

Commented [PP1]: oneMDS?

3.2 Data Models

The below tables give the tables required specifically for integration with SFSF

3.2.1 Existing Data Model Changes

3.2.1.1 CostCenters

- Two entities need to be added as 1:n compositions to the base entity CostCenters therefore the CostCenters entity should have 1:n references to these entities similar to this
  - CostCenterValidity
  - CostCenterAttributes
- CostCenterValidity
  - Use aspect workforceCommon.temporalComposites to define the keys similar to this
  - Move isValid attribute to this entity
- CostCenterAttributes: rename Attributes to CostCenterAttributes
  - Use aspect workforceCommon.temporalComposites to define the keys similar to this
- Remove isBlockedForPrimaryPosting and isBlockedForSecondaryPosting from the attributes entity as this is not needed for RM.

3.2.2 OneMdsDeltaToken

Commented [PP2]: Rename to OneMdsDeltaToken

Commented [KK3R2]: Ok. We will change.

This table is required to store the last oneMDS delta token attributes from LOG API response.

Name	Description	Datatype	Primary	Foreign
DELTATOKEN		STRING(20)		
ENTITYNAME		STRNG(50)	X	

Commented [PP4]: We also need the entity name here to indicate either cost center or workforce as delta token would be different for each of them

Commented [KK5R4]: Yes. Cost center is also part of integration. I will add entityName column attribute.

Commented [PP6R4]: Key?

Commented [KK7R4]: Delta Token will be unique value.

3.2.3 *ReplicationFailures*

This table is required to store replication failure details.

Name	Description	Datatype	Primary	Foreign
VERSIONID	Key	UUID	X	
INSTANCEID	Key	UUID	X	
EXTERNALID		NVARCHAR(100)		
EVENT		NVARCHAR(20)		
REPLICATIONTYPE		NVARCHAR(50)		
REPLICATIONERRORMESSAGE		*REPLICATIONERRORMESSAGES*		
ERRORPARAM1		NVARCHAR(100)		
ERRORPARAM2		NVARCHAR(100)		
ERRORPARAM3		NVARCHAR(100)		
ERRORPARAM4		NVARCHAR(100)		
REPLICATIONERRORMESSAGE		*REPLICATIONFAILURESTATUS*		

3.2.4 *ReplicationErrorMessages*

Name	Description	Datatype	Primary	Foreign
CODE	Key	NVARCHAR(12)	X	
ERRORMESSAGE		NVARCHAR(255)		
DESCRIPTION		NVARCHAR(255)		

3.2.5 *ReplicationFailureStatus*

Name	Description	Datatype	Primary	Foreign
CODE	Key	INTEGER	X	
DESCRIPTION		NVARCHAR(255)		

3.2.6 AvailabilityReplicationSummary

Name	Description	Datatype	Primary	Foreign
RESOURCEID	Key	UUID	X	
WORKFORCEPERSONEXTERNALID		NVARCHAR(100)		
COSTCENTERID		UUID		
WORKASSIGNMENTSTARTDATE		NVARCHAR(20)		
WORKASSIGNMENTENDDATE		NVARCHAR(20)		
WORKASSIGNMENTEXTERNALID		NVARCHAR(100)		
NOOFRECORDPROCESSED		INTEGER		
NOOFRECORDSFAILED		INTEGER		
NOOFRECORDSPASSED		INTEGER		
AVAILABILITYSUMMARYSTATUS_CODE		INTEGER		
AVAILABILITYSUMMARYSTATUS		*AVAILABILITYSUMMARYSTATUS*		CODE
TODELIVERYORG		*HEADERWITHDETAIS*		COSTCENTERID
AVAILABILITYREPLICATIONWERROR		AVAILABILITYREPLICATIONERROR		RESOURCE_ID
TPRPOFILEDATE		PROFILEDATABUILDER		COSTCENTERID
AVAILABILITYSUMMARYSTATUS		*AVAILABILITYSUMMARYSTATUS*		CODE

3.2.7 *AvailabilitySummaryStatus*

Name	Description	Datatype	Primary	Foreign
CODE	Key	INTEGER	X	

3.2.8 *AvailabilityReplicationError*

Name	Description	Datatype	Primary	Foreign
RESOURCEID	Key	UUID	X	
STARTDATE	Key	NVARCHAR(20)	X	
S4COSTCENTERID		NVARCHAR(10)		
WORKASSIGNMENTEXTERNALID		NVARCHAR(100)		
ERROR_DESC		NVARCHAR(255)		
CSVRECORDINDEX		*NVARCHAR(10)*		
INVALIDKEYS		NVARCHAR(10)		

3.2.9 *AvailabilityReplicationStatus*

Name	Description	Datatype	Primary	Foreign
RESOURCEID	Key	UUID	X	
STARTDATE	Key	NVARCHAR(20)	X	
S4COSTCENTERID		NVARCHAR(10)		
WORKFORCEPERSONEXTERNALID		NVARCHAR(100)		
STATUS		NVARCHAR(10)		

3.3 Integration Strategy

3.3.1 Employee Basic Data:

Employee basic data would be replicated into RM from oneMDS using the Log API. Systems should poll this API to retrieve change events for specific master data objects. To retrieve updates for a specific entity type a system has to send a GET request to the respective endpoint. More details to the API are here

3.3.1.1 Source to Target Mapping

The below section gives the mapping between the source and target system for the fields involved in the integration along with mapping logic if any.

RM Entity	RM Field Name	ONEmds entity	ONEmds field name	Comments
Headers	ID	WorkforcePerson	ID	This is the personUUID from successfa
WorkforcePersons	ID	WorkforcePerson	ID	
WorkforcePersons	externalID	WorkforcePerson	externalId	
SystemOfRecordKey	systemOfRecordId	SystemOfRecordKey	systemOfRecordId	

SystemOfRecordKey	systemOfRecord-WorkAssignmentId	SystemOfRecordKey	systemOfRecord-WorkAssignmentId	
SystemOfRecordKey	workAssignmentExter-nalId	SystemOfRecordKey	workAssignmentExter-nalId	
SystemOfRecordKey	isCurrent-SystemOfRecord	SystemOfRecordKey	isCurrent-SystemOfRecord	
ProfileDetails	valid_from	ProfileDetails	valid_from	
ProfileDetails	valid_to	ProfileDetails	valid_to	
ProfileDetails	firstName	ProfileDetails	firstName	This comes from profileDetail->conter and map
ProfileDetails	middleName	ProfileDetails	middleName	This comes from profileDetail->conter and map
ProfileDetails	lastName	ProfileDetails	lastName	This comes from profileDetail->conter and map
SourceUserAccount	userName	SourceUserAccount	userName	
WorkAssignment	startDate	WorkAssignment	startDate	
WorkAssignment	endDate	WorkAssignment	endDate	
WorkAssignment	externalId	WorkAssignment	externalId	
WorkAssignment	isContingentWorker	WorkAssignment	isContingentWorker	
WorkAssignment	workAssignmentID	WorkAssignment	id	Instead of tempId, RM has ID field wh successfactors. We will introduce a ne oneMDS. **Important Note:** During integration a **created for a unique combination of**
WorkAssignmentDetails	valid_from	WorkAssignmentDetails	valid_from	
WorkAssignmentDetails	valid_to	WorkAssignmentDetails	valid_to	
WorkAssignmentDetails	isPrimary	WorkAssignmentDetails	content->isPrimary	
WorkOrderDetails	valid_from	WorkOrderDetails	valid_from	
WorkOrderDetails	valid_to	WorkOrderDetails	valid_to	
WorkOrderDetails	supplier	WorkOrderDetails	supplier->externalId	
JobDetails	valid_from	JobDetails	valid_from	
JobDetails	valid_to	JobDetails	valid_to	
JobDetails	legalEntityexternalId	JobDetails	legalEntity->externalId	
JobDetails	supervisorWorkAssign-mentExternalId	JobDetails	supervisor-WorkAssignment->ex-ternalId	
JobDetails	jobExternalId	JobDetails	job->externalId	
JobDetails	jobTitle	JobDetails	jobTitle	
JobDetails	costCenterExternalId	JobDetails	costCenter->id	
JobDetails	country	JobDetails	country->code	
JobDetails	workingDaysPerWeek	JobDetails	workingDaysPerWeek	
JobDetails	workingHoursPerWeek	JobDetails	workingHoursPer-Week	
JobDetails	eventSequence	JobDetails	eventSequence	
JobDetails	eventCode	JobDetails	event->code	
JobDetails	eventReasonCode	JobDetails	eventReason->code	
JobDetails	orgUnitExternalId	JobDetails	orgUnit->externalId	

JobDetails	superOrdi-nateOrgUnit1ExternalId	JobDetails	superOrdi-nateOrgUnit1->exter-nalId	
JobDetails	superOrdi-nateOrgUnit2ExternalId	JobDetails	superOrdi-nateOrgUnit2->exter-nalId	
Phone	code	Phone	usage->code	Bring only workphone and mobile pho The way to do this would be to look at phone by the customer. For example ht pand=phoneTypeNav
Phone	number	Phone	number	
Phone	isDefault	Phone	isDefault	
Phone	country	Phone	country->code	
Email	address	Email	address	
Email	isDefault	Email	isDefault	
Email	code	Email	usage->code	Bring only official business email from email classified as business by the cust tors/odata/v2/PerEmail?$expand=emai uration table.
PrivateAddress	valid_from	PrivateAddress	valid_from	All private attributes to be confirmed i
PrivateAddress	valid_to	PrivateAddress	valid_to	
PrivateAddress	usage->code	PrivateAddress	usage->code	
PrivateAddress	country->code	PrivateAddress	country->code	
PrivateAddress	primaryRegion->code	PrivateAddress	primaryRegion->code	
PrivateAddress	secondaryRegion->name	PrivateAddress	secondaryRegion-->name	
PrivateAddress	district->name	PrivateAddress	district->name	
PrivateAddress	town->name	PrivateAddress	town->name	
PrivateAddress	postCode	PrivateAddress	postCode	
PrivateAddress	street->name	PrivateAddress	street->name	
PrivateAddress	streetPrefix1	PrivateAddress	streetPrefix1	
PrivateAddress	streetPrefix2	PrivateAddress	streetPrefix2	
PrivateAddress	streetSuffix1	PrivateAddress	streetSuffix1	
PrivateAddress	streetSuffix2	PrivateAddress	streetSuffix2	
PrivateAddress	houseNumber	PrivateAddress	houseNumber	
PrivateAddress	houseNumberSupplement	PrivateAddress	houseNumberSupple-ment	
PrivateAddress	floor	PrivateAddress	floor	
PrivateAddress	door	PrivateAddress	door	
PrivateAddress	careOf	PrivateAddress	careOf	
PrivateAddress	alternative_deliv-eryServiceType	PrivateAddress	alternative_deliv-eryServiceType	
PrivateAddress	alternative_deliv-eryServiceQualifier	PrivateAddress	alternative_deliv-eryServiceQualifier	
PrivateAddress	alternative_deliv-eryServiceIdentifier	PrivateAddress	alternative_deliv-eryServiceIdentifier	
Resource.Headers	ID			Mapped to every generated workassign
Resource.Headers	type_code			To be decided

3.3.1.2 ID Mapping

Since there are several GUIDs and IDs in EC and the inherited DMA models of worker which will be used in oneMDS, this is the summary of the ID mapping that exists between EC and oneMDS.

ONEmds	EC Mapping	Comment
WorkforcePerson.id	PerPerson.perPersonUuid	Immutable
WorkforcePerson.externalId	PerPerson.personIdExternal	Mutable
WorkAssignment.id	EmpEmployment.userId	Do not use as of now
WorkAssignment.externalId	EmpEmployment.assignmentIdExternal	Needs to be enabled by a switch. Mutable Net new customers can generate this value as PERNR in ERP or S/4 (in that case immutable)
SystemOfRecordKeys.systemOfRecordWorkAssignmentId	EmpEmployment.userId	Can be used for key mapping. Every Assignment in EC will have a separate row in SystemOfRecordKeys
SystemOfRecordKeys.workAssignmentExternalId	EmpEmployment.assignmentIdExternal	
SystemOfRecordKeys.systemOfRecordId	Company/Tenant	Logical system

Impacts on RM:

Workassignment table ID generation Logic:
In the workassignment table RM has ID field which is the generated guid key field which is similar to the tempID field in workforce Workassignment table of oneMDS model. The id field in this table in oneMDS is mapped to userid of successfactors. We will introduce a new field called workAssignmentID (string 100) and map it to this ID field of oneMDS.
Important Note: During integration a new GUID in workassignment table as the key(in Id field) in RM **should only created for a unique combination of WorkforcePersonId and ID from oneMDS as this is the logical key.**

3.3.2 Employee Availability

Employee availability data integration will be handled using new application with upload functionality.

3.3.2.1 Pre-requisites

The pre-requisite to upload availability CSV is that the basic data integration for the employee should be completed.

3.3.2.2 Approach

The table *AvailabilityReplicationSummary* will have the workAssignmentId field. This table will be initially filled when employee basic data replication for the resource with resource_id same as "ID" field (key) in work assignment table and workAssignmentId field same as workAssignmentId field.

Any failures during upload process will be stored into *AvailabilityReplicationError* table.

3.3.2.3 Availability Service API Details

I. ***Availability File Upload Service***.

The API Defines an HTTP endpoint to upload Availability data as a CSV file

Resource	**/AvailabilityFileUploadService**		
HTTP Operation	**POST**		
Return Type	**ImportResult**		
Authorization	**Token Based Authentication**		
Throws	**IOException**		
Request Parameters	**Mandatory/Optional**	**Type of Value**	**Default Value**
file	Mandatory	MultipartFile	
costcenter	Mandatory	String	

II. Availability File Download Service.

The API Defines an HTTP endpoint to download Availability summary data as a CSV file.

Resource	**/AvailabilityFileDownloadService**		
HTTP Operation	**GET**		
Return Type	**StreamingResponseBody**		
Authorization	**Token Based Authentication**		
Request Parameters	**Mandatory/Optional**	**Type of Value**	**Default Value**
startDate	Mandatory	String	
endDate	Mandatory	String	
costCenter		String	
workForcePersonId		String	
defaultWorkinghours		Integer	0
defaultNonWorkinghours		Integer	0

The API date format:

- startDate : Date type Format: YYYY-MM-DD e.g., 2016-01-01
- endDate : Date type Format: YYYY-MM-DD e.g., 2016-01-01

Example
http://<host>:<port>/AvailabilityFileUploadService?costcenter=CCIN
http://<host>:<port>/AvailabilityFileDownloadService?startdate=2020-01-01&enddate=2021-03-31&costcenter=CCIN&workforcepersonid=&working-hours=8&nonworkinghours=0

3.4 Implementation Details

3.4.1 Design Considerations

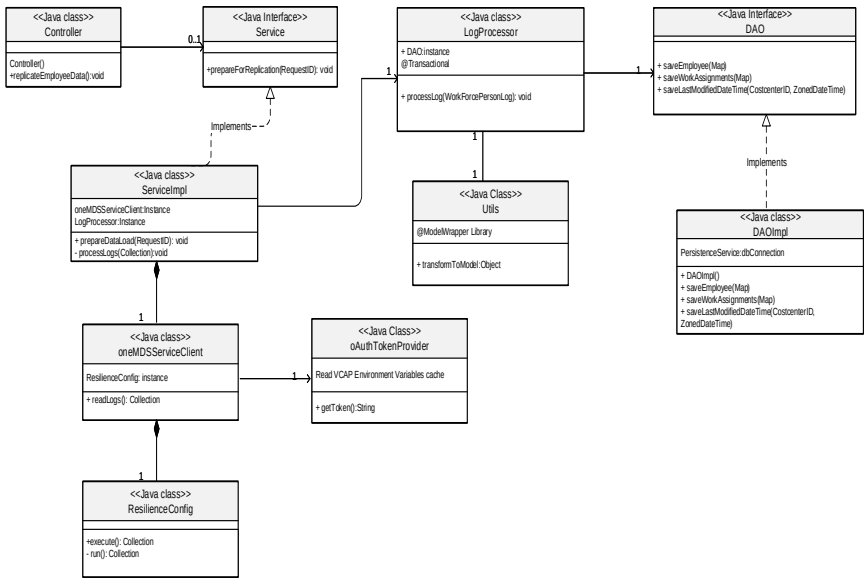
- Optimal memory footprint in terms of not keeping too much data in main memory during the integration data processing should be followed. For example 100KB of data at a time would be a good number to begin with, this should be achieved with proper

page size, limit hint parameters for the initial and delta load and the KPIs needs to adjusted and tuned during the performance testing.

- Sequential execution of data with proper pagination will be followed to begin with. Concurrent execution can be looked at a later phase.
- Proper error handling and error monitoring will be in place so that the status of replication is visible and appropriate action to correct the errors is taken care <<Open Point>> how does re-processing happen for errored records
- The design will need to ensure resilience and fault tolerance. For example what should happen when oneMDS service is down or when successfactors system is not reachable? So that recovery is quick with minimal impact.
- The duration of the update lifecycle should be kept very short where we collect the updates and perform the commit. So depending on the page size where we process a set of records (lets 100-300), we collect these records, commit and then move to the next set of records which we pull.

3.4.2 oneMasterDataService : Basic Data

Class Diagram



Class Level Details

Commented [PP8]: We don't need to repeat the design for availability and cost center but in general we create two subsections for each of them and refer to the relevant classes in the section for specifics

Commented [KK9R8]: Classes are different for oneMDS availability and thier implementation is different.

Commented [KK10R8]: I will update the class diagram for availability seperately once finalized.

1. *Controller* class

   The controller class will get the request, according to the request *replicateEmployeeData*() method of controller will be called and processing will be done. The controller calls the service method *prepareForReplication()*. with RequestID (correlationID) will be unique for each request.
   This class will be responsible to create Scheduler configuration object and returns the HTTP response as ACCEPTED.

2. *Service* class

   Logical part of the application.
   This class has DAO object reference which is autowired automatically as the class is loaded.
   The *prepareForReplication* () method will help us to identify the process[initial/Delta] to be started. This class connects to table *EmployeeReplicationInfo* to read *delta token.*

   This class implements the pagination concept using the API response attribute "hasMoreElements" and accordingly triggers call to oneMDS.

   The method *processLogs*() will process the records fetched by initial/Delta load process.
   .
3. *oneMDSServiceClient*

   A client implementation providing blocking API. This class is responsible to initiate a connection to underline oneMDSService. The *oAuthTokenProvider* bean autowired to configuration to provide authorization for the API.

   JWT Exchange details:

   For each tenant subdomain, client credentials flow will be initiated with SaaS user subdomain. The client_id and client_secret will be read from one-mds clone service instance. These values will be read using *java-cfenv* library.

   Using above generated access token, we will be invoking ONE MasterData service LOG API to replicate workers. The access token must be used in header as Authorization: Bearer

4. ResilienceConfig

   This class will have implementation for CircuitBreaker and Retry patterns and will be configured for oneMDSService call.

   This class will use Resilience4J third party library to handle resiliency.

   In case when oneMDS service not available a default fallback will be returned.

Commented [PP11]: What is blocking API?

Commented [KK12R11]: Send an HTTPS request to oneMDS service and waits for the response. It will block the process until it gets the response from service.

5. LogProcessor

   This class is responsible for:

   a) Converters [oneMDS response to RM models]
   b) Build RM objects
   c) Foreign Key mapping
   d) GUID generation
   e) Persisting of employee details

6. *DAO class*

   This class follows DAO pattern and is used to persist following model objects:

   a) Employee Header
   b) EmployeeReplicationInfo.

7. *Utility class*

   Helper class to provide utility functions like model conversion from dto-entities vi-ceversa.

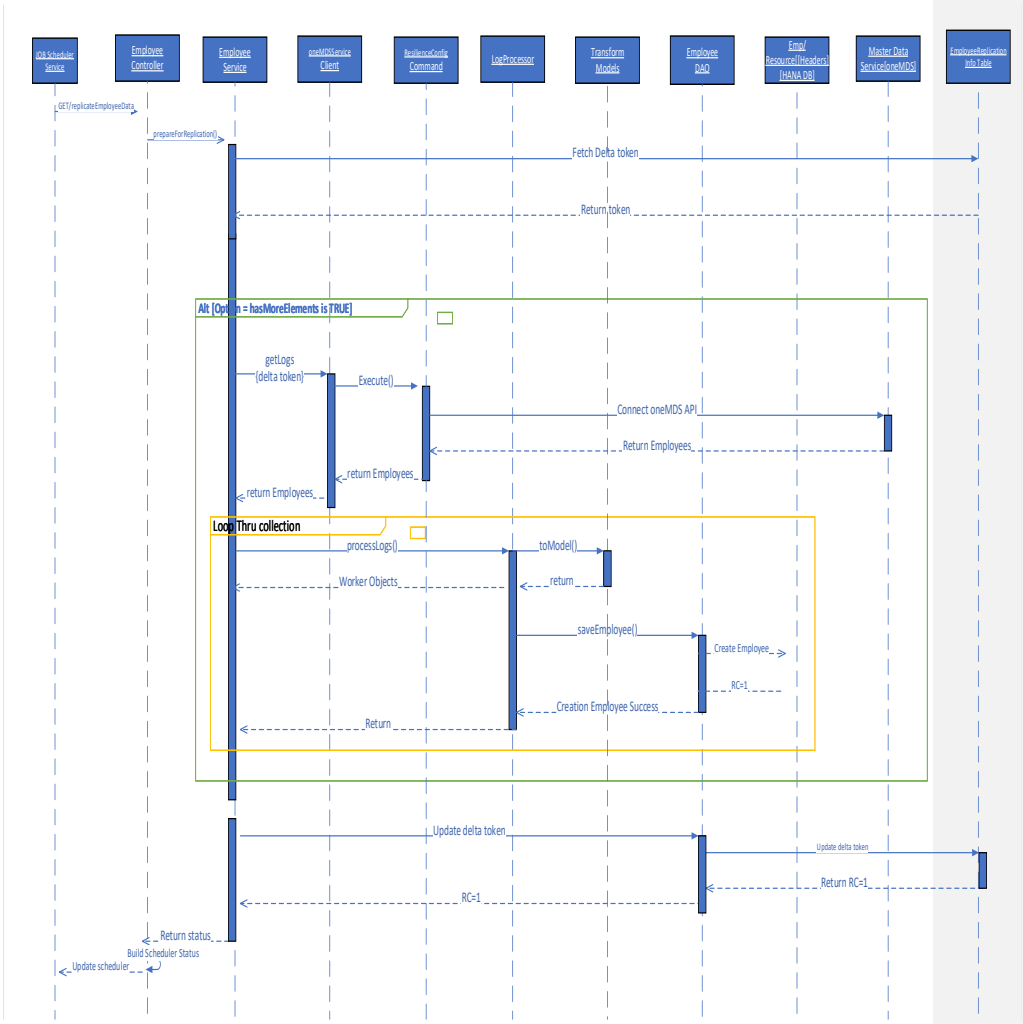
8. *oAuthTokenProvider class*

   This class provides OAuth2 Bearer tokens to authenticate against ONEmds. When a token is acquired it should be cached until it expires.
   The necessary values to retrieve oneMDS url, client id and client secret must be read from VCAP.

Sequence Diagram

The below figure shows a high-level process flow for initial and delta loading of employees.

Replication Process



DataFlow Diagram

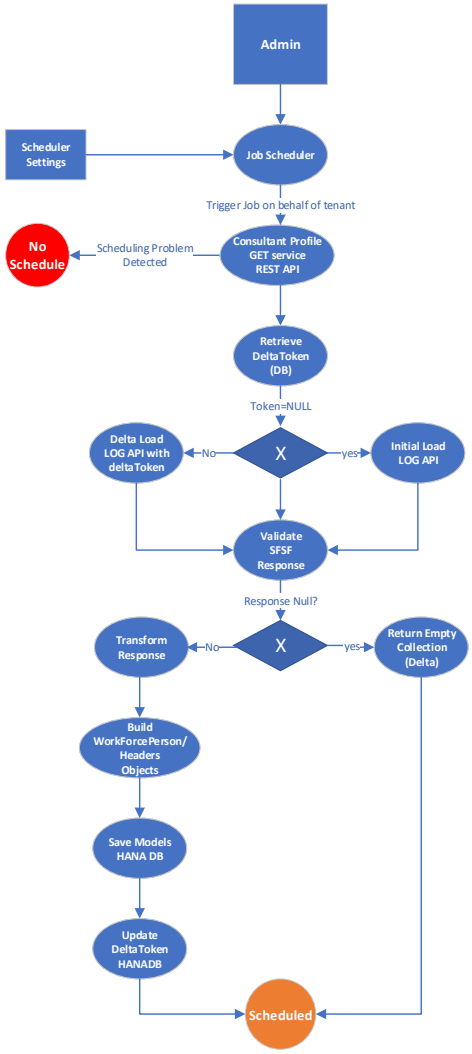


Fig: Dataflow Diagram

- As part of solution, we will be using PULL-Based strategy to replicate employee information from oneMDS service. Pull-based describes a client-run scheme. Clients make requests, and the data is served up immediately.

- Retrieve tenant details using SaaS registry API. For each tenant, fetch Identity Zone information (SAAS subdomain).
- SAP Cloud Platform JOB Scheduler service is scheduled to call Consultant Profile service. Following steps are performed during implementation:
    a) Retrieve tenant details using SaaS registry API. For each tenant, fetch Identity Zone information (SAAS subdomain).
    b) As described above, Job scheduler will be configured to replicate employees.
    c) Create a job specific for a SaaS tenant. We be using client credentials flow with SaaS user subdomain. We need to replace the URL subdomain with the corresponding SAAS subdomain for which the token is requested.
    d) Using the access token received, we be invoking Job Scheduler REST API to Create Job. The access token has to be used in header as Authorization: Bearer
- Admin Launchpad will help the tenants to provide configuration details like cost center, employees etc. These details will be available for integration process through *Service_Org* table. Following is the github link

  https://github.wdf.sap.corp/Cloud4RM/Consultant-Profile/blob/master/db/cds/core/organization.cds

  Firstly, the Headers entity would have the list of service organization and its associated cost centers would come from the Details entity with unitType as 'CS'
- Employee replication is carried out based on cost center the Employee belongs. ONEmds provides a data filtering engine to restrict the events or the data contained within the events that a system receives from the Log API. Section below defined more details on Data Filtering.
- The application will store last successful deltaToken employee replication details in *OneMDSDeltaToken* table. Read delta token value from *OneMDSDeltaToken* table. If value is NULL, initial Load process will be called, otherwise Delta Load process is called.
  Pls refer table structure under Data Models section 3.4.
- The Filters used as part of Pull are:
    - Employee Cost Center and
    - Not a contingent worker
    - Status in Job Details -> WorkforcePerson. workAssignments. JobDetails.status.code
        - The status field of job details indicates the employee status. It might have the values e.g. active, inactive, terminated, unpaid leave, dormant etc.
    - Each of these status codes in the EmploymentStatus HR Status field mean the following

Commented [PP13]: @Khamitkar, Kirankumar I understand the cost center and contingent worker field by what are these two other fields in our data model. What do you mean by EmployeeRecordAvailable
@Amalraj, Raymond -> can we please finalize the list of filters that we need, we need to then reach out to Workforce team to see what is supported, what is not. In addition to the fields what kind of validity dates for the entities are we targeting, are we setting a filter where validity date is from the beginning of the current year

Commented [KK14R13]: @P, Praveen EmployeeHiring process completed : Hiring process completed, the value filtered in SFSF is as below:

"employmentNav/hiringNotCompleted eq false"

and Employee Record available is to the check is EC record. The filter applied in SFSF as below:

"employmentNav/isECRecord eq true"

These filters were used in MDW also. We just are referring the same.

Commented [AR15R13]: i had put the filters part in this epic https://sapjira.wdf.sap.corp/browse/MODULAREPPMRE-LEASESCP-795

Commented [KK16R13]: @P, Praveen :Complete filter used to fetch from SFSF as below:

employmentNav/hiringNotCompleted eq false and employmentNav/isECRecord eq true and employmentNav/isContingentWorker eq false.

Commented [AR17R13]: The requirement for data filters identified so far are as follows:
* Allow a filter to be set for replicating only employees without contingent workers or also including contingent workers.
Apply filters on
* Set of Cost centers relevant for resource management. This can be obtained from service Org master data which has links to relevant cost centers.
* Set of company codes relevant for resource management.
* Work Assignment External ID (may be used as personnel numbers by customers)

Commented [AR18R13]: EC record, employee hiring may not be relevant now as they were filters for replicating from SFSF to OneMDS. WE consume directly from OneMDS. Hence they are not relevant for us is my understanding

Commented [PP19R13]: @Khamitkar, Kirankumar there are no fields like those in oneMDS as per my understanding, instance or segment filters can only be applied on the current model entities and attributes, so I don't think these filters hold good here. Please check and remove.

Commented [PP20R13]: @Amalraj, Raymond ->
1)I don't think we can apply filters anymore like before with S/4 cost centers as this cost center information is not part of oneMDS workforce model, it can only be applied on cost center guid and you have to post this request against another API, so you need know the GUID beforehand is my understanding. We need to get into details of such requirements with Frank and team.
2) We need to request them to add additional filters as per the email sent by Sagar this is the list currently supported, some of them instance, some segment active
`WorkforcePerson. workAssignments.isContigent-Worker`

Commented [KK21R13]: @P, Praveen Removed

A	Active
D	Dormant
F	Furlough
O	Discarded
P	Paid Leave
R	Retired
S	Suspended
T	Terminated
U	Unpaid Leave

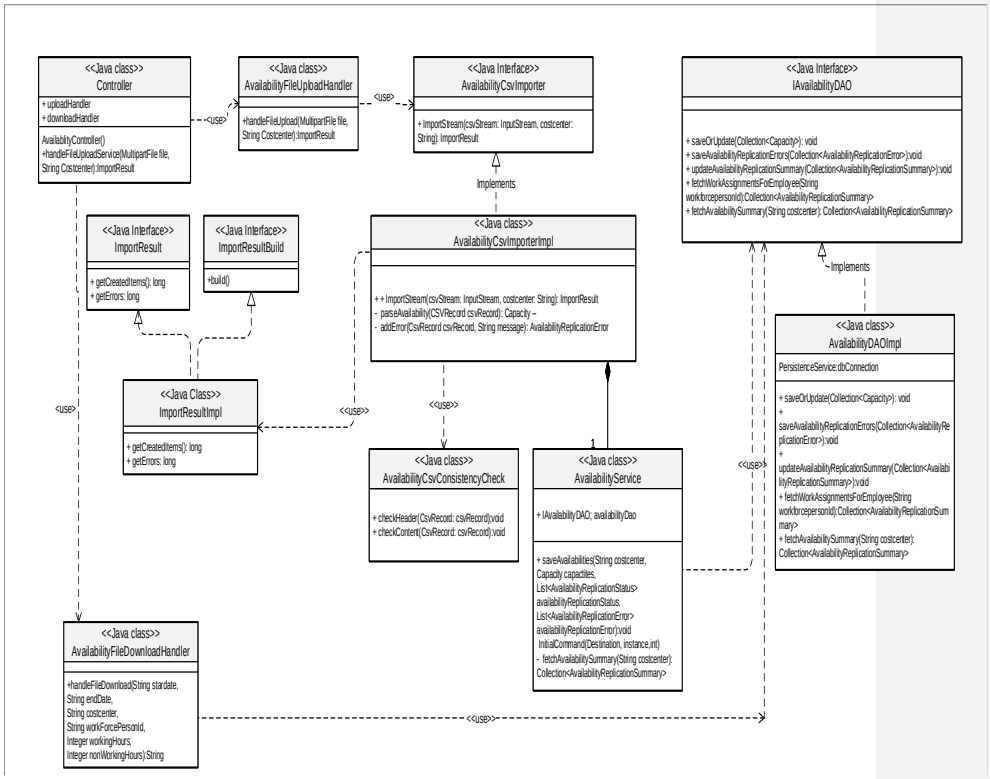
All these above criteria to be triggered while using Filtering API.

- Validate response process, validates Query Response for not null and checks mandatory attributes like validFromDate, validToDate, externalID etc as part of response.
- For successful responses, transformation is carried out at entity and sub-entity levels using spring framework converter interfaces and accordingly, RM based models are populated with relevant information.
- During objects build phase, *WorkForcePersons* and Headers models are populated with the unique GUID's and accordingly associated with Foreign Keys across entity and sub-entities.
- Next, the models above are replicated to managed HANA DB and accordingly the last successful sync date time is recorded into DB. We will be using CAP Persistence Service to insert data is into tables directly using CDS4J UPSERT functionality.
- After successful replication of employees, the delta token value from *EmployeeReplicationInfo* needs to be updated with latest delta token retrieved from response.
- Exceptional Handling will be done at each employee level. If replication fails for an employee, the employee details along with reason for failure is logged in the *EmployeeReplicationFailures* table.

Pls refer table structure under Data Models section 3.4

3.4.3 Availability Data Upload/Download functionality

Class Diagram



Note: The service class will host two API for implementation. Respective implementation differ for availability and basic data.

Class Level Details

1. *Controller* class

   The Availability controller will host two endpoints.

   1/ Handle POST request to upload CSV details.
   2/ Handle GET request for downloading CSV data.

The Availability controller class will get the request and according to the request *respective* methods of controller will be called and processing will be done.

2. *AvailabilityFileUploadHandler* class

   This class has service object reference which is auto-wired automatically as the class is loaded.
   The *handleFileUpload* () method validates the size of multipart file array and in case size is 1, the uploaded file is processed.

3. *AvailabilityFileDownloadHandler* class

   This class has DAO object reference which is auto-wired automatically as the class is loaded.
   The availability replication summary data is populated as part of employee basic data replication. The *handleFileDownload* () method fetches the availability replication summary view details from database for given costcenter or workforcepersons details. It validates the work assignments based on assignment start and end dates and renders the details as stream of strings.

4. *AvailabilityCSVImporterImpl class*

   This class reads the input stream as CSV file. After checking its consistency, the data is extracted and stored in resource capacity table.

   An ImportResult object is created to signal which capacity were created successfully and which failed.

5. *AvailabilityDAO class*

   This class follows DAO pattern and is used to persist and read following model objects:

   a) Capacity
   b) AvailabilityReplicationSummary.
   c) AvailabilityReplicationErrors.

6. *ImportResult class*

   Class holds details of data returned as a result from an import containing successful items, failed items.

7. *AvailabilityService class*

   Logical part of the application.
   This class has DAO object reference which is auto wired automatically as the class is loaded.

   This class implements in a transactional way to process and persists the database models.

Pls refer table structure under Data Models section 3.2

3.5 Other Topics

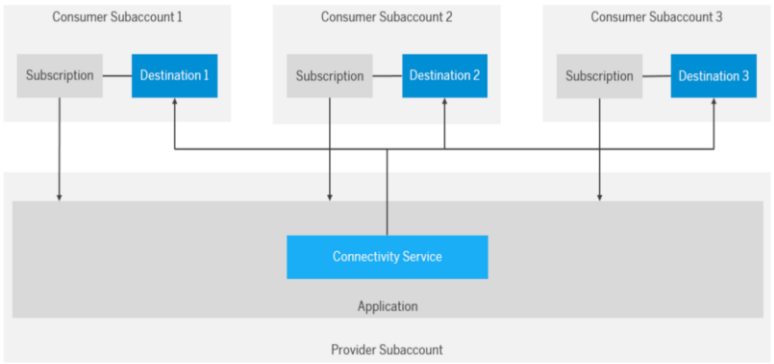
3.5.1 Multitenancy in the Connectivity Service

SAP Cloud Platform Connectivity provides two services for the Cloud Foundry environment, the Connectivity service and the Destination service.
Using the Destination service, you can retrieve and store the technical information about the target resource (destination) that you need to connect your application to a remote service or system.

Note:
We use subscriber specific Destination handling to read destinations defined in customer sub-account. We as part of integration will read subaccount destinations by using below API form destination service.

https://destination-configuration.cfapps.sap.hana.ondemand.com/destination-configuration/v1/subaccountDestinations

Subscriber Specific Destination:



For a destination associated with a subscriber subaccount:

1. Checks if the destination is available on the subscription level.
2. If there is no destination found, it Searches the destination on subaccount level.

3.5.2 *oneMDS* Data Filtering

ONEmds provides a data filtering engine to restrict the events or the data contained within the events that a system receives from the Log API.

A filter has to be registered for a tenant (*serviceInstanceId*) via a separate API. After registering a filter for a system, for all following requests of this system to the Log API, the response

of the Log API is based on the conditions of the activated filter. In ONEmds, filters are represented as CQL queries.

- Instance filters can be expressed in the where clause of a query.
- Segment filters can be expressed in select clause
- To filter on temporal fields, it is necessary to access the internal structure of temporal fields.

Sample code snippet of filter CQL Queries

select from sap.odm.workforce.WorkforcePerson {* ,(select from temporal(from: jobDetails) { * } where (content.country.code = 'US' or content.country.code = 'DE') and (content.legalEntity.externalId = '100'))

For more details following is the link:
https://github.wdf.sap.corp/pages/re-wunder/ONEmasterdata/index.html#user-guide-data-filtering

3.5.2.1 Filter Process in RM

There are two options for filters to be set

Option A: Set the filter before employee replication via tenant specific job

Following filters would need to be set in oneMDS for employee basic data replication.

Filter Fields	**Description**	**Include/Exclude**	**Type**	**Value**
WorkforcePerson. workAssignments. JobDetails.status.code	**Job Status**	**I**	**Instance**	**A, F,P,S,U**
WorkforcePerson.workAssignments. startdate	**Work assignment start date**	**I**	**Instance**	**>= 01.01.2020**
WorkforcePerson.WorkAssignment.JobDetails.Cost Center	**Cost Center**	**I**	**Instance**	**Derived at runtime**
WorkforcePerson.WorkAssignment.JobDetails.Cost Center	**Cost Center**	**I**	**Segment**	**Derived at runtime**
WorkforcePerson.phones.usage.code	**Phones**	**I**	**Segment**	**Work (“B”) and Mobile Phones (“C”)**
WorkforcePerson.emails.usage.code	**Email**	**I**	**Segment**	**Business email (“B”)**

The tenant specific background job would first invoke the filters as per the following before the employee basic data replication is triggered

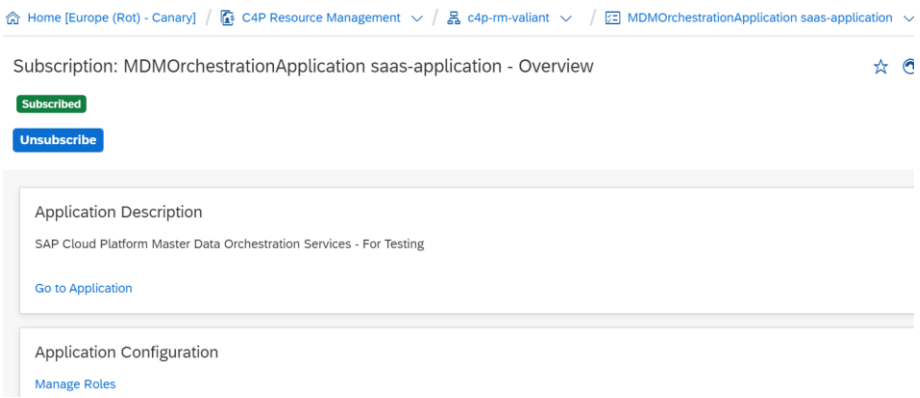
- Get the list of cost centers maintained in table com.sap.resourceManagement.organization.Details for the unitType = “CS” (A)
- Get the list of cost centers from table MDIReplicationFilterInfo for the REPLICATION_TYPE = “Employee” (B)
- If A is not equal to the list in B this means the cost centers in RM have changed and therefore the filter need to be updated for the new cost centers.
- For this list of cost centers obtained in step 1 (A) find the corresponding Costcenter GUIDS from table com.sap.resourceManagement.organization.CostCenters (field ID) for the list of cost centers obtained in previous step (against field LocalIdS4-costCenterId)
- Apply the filters as described in the table above for the service instance id. The assumption is that customers have activated the filter feature
    - After activating the filter feature, filters can be deployed with an http POST request to the mdo-api.
    - Use https://one-mds-mdo.cfapps.sap.hana.ondemand.com/v0/orchestration/odata/v4/mdo.replication as the url.
    - The payload should look like in this sample.json
    - the name of the entity is determined based on
        - the BusinessObjectType_objectTypeCode value = 167 representing workforce
    - the instance filters are declared in To_ApplicationScope and the segment filters in To_DistributionScope
    - the understood operators for the filters are LT, LE, GT, GE, EQ and BT
    - positive filters (sign is true) with the same ValuePath are conjuncts that produce a CNF for all positive filters
    - negative filters (sign is false) are conjuncts regardless of ValuePath and their overall negation is AND-added to the CNF
    - This is done for the id of the service instance (= system) for which the filter should be activated
    - you must declare the AttributeDataType for each of the filters: if you don’t then the POST request will return 400
- If successful Update the table MDIReplicationFilterInfo with FILTERLASTMODIFIEDDATETIME as the current timestamp for REPLICATION_TYPE = “Employee”
- Trigger the employee replication
    - If filter application fails, update job log do not update MDIReplicationFilterInfo and trigger the employee replication, retrigger the process in the next job run after fixing the errors if any

Option B: Set the filters manually via the UI application provided by MDI

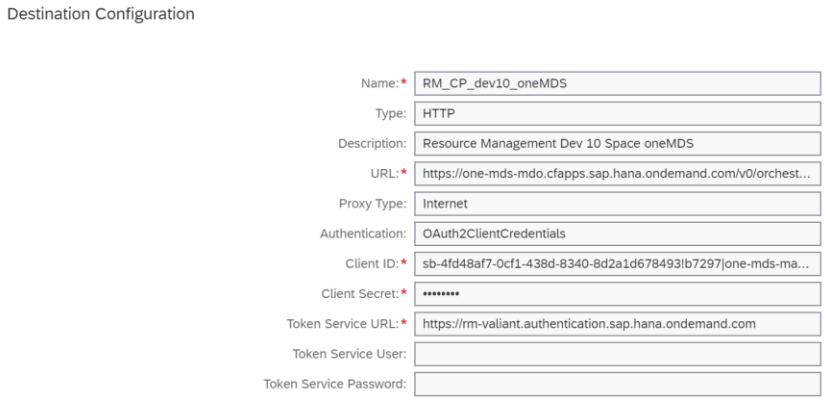
Pre-requisites: A global and account and sub-account is already created in SCP-CF via the cloud platform cockpit and oneMDS service is available in the sub-account and space. Further a service instance to the oneMDS service has also been created.

Steps:

- In the control center add the MDO application SAP Cloud Platform Master Data Orchestration Services to the global account and assign the requisite quota
- In the cloud platform cockpit for the global account and subaccount, subscribe to the application



- Navigate to the subaccount in SAP Cloud Platform Cockpit.
- Navigate to Connectivity Destinations.
- Choose New Destination and enter the details as follows:
- Name: <Name for the new destination>. For example, MDO_DEST.
- Type: HTTP
- URL: https://one-mds-mdo.{data center dns suffix}/v0/orchestration/odata/v4/mdo.orchestrationAdmin
- Proxy Type: Internet
- Authentication: OAuth2ClientCredentials
- Please use information from Service Key created in previous step
- Choose New Property from Additional Properties. Enter the MDOConsumer property with value true.
    - Choose New Property from Additional Properties. Enter the MDOONEmds property with value true.



Summary:

Between options A and B, option A is the preferred option for RM as the control of the filters being set will be with RM. However after checking with MDI team it was decided to go with option B as recommended way of setting filters. The filters that would be set by the customer and the process behind activating the jobs after setting the filters would need to be documented for the customer.

3.5.3 oneMDS Change Events when consuming the Log API

Master data objects have a persistent identity but otherwise change over time. For example, a cost center with a particular ID can be renamed. Every time a master data object changes, the ONEmds service captures the new values for all properties of that object as a change event with a unique version identifier. Change events are always replicated atomically, so the change event is the primary provider of consistency.

There are different kinds of change events to notify receiving systems about what happened to master data:

- **Created.** Sent when a master data object instance is first replicated to a receiving system by ONEmds.
- **Updated.** Sent when a master data object instance changes that was already replicated to the receiving system by ONEmds in an earlier "created" event.
- **Deleted.** Sent when a master data object instance that was previously replicated to the receiving system by ONEmds has been deleted.
- **Excluded.** Sent when a master data object instance that was previously replicated to the receiving system by ONEmds will no longer be replicated due to filters. This can happen because the properties of the master data object instance changes or because the filter configuration changes.
- **Included.** Sent when a master data object instance that was previously "excluded" will be replicated again. This can happen because the properties of the master data object instance changes or because the filter configuration changes.
- **Rejected.** Sent when a change request by the receiving system was rejected by ONEmds. For example, this can happen due to schema validation failures or due to optimistic locking.

The "created", "updated", "included", and "merged" events contain the new state of the master data object instance after the event (all properties, not just the changed properties). The "deleted" and "excluded" events only contain the global UUID of the affected master data object instance, and "rejected" events contain no information about the affected master data object instance. All events can contain the change token of the change request that led to the event, but the change tokens are only included if the same system that sent the change request also receives the event. Similarly, "rejected" events are only exposed to the system that sent the change request that was rejected, while the other events are exposed to all receiving systems.
ONEmds exposes these master data events to receiver systems via the Log API. Systems fetch events from the master data event log with HTTPS GET requests. Authenticity, confidentiality, and data integrity of the communication with the Log API is provided via TLS. Authentication of receiver systems is provided by OAuth2 with JWT tokens.

Each HTTP response can contain multiple events. If something goes wrong with processing a log request, ONEmds reacts with an error response. Systems can include a delta token and a limit hint in their requests to the Log API. ONEmds will use the delta token to not send events again that the system has already received in an earlier call. ONEmds will use the limit hint to control how many events are sent with the response. ONEmds will include a new delta token and a more data hint in the response. The new delta token has to be stored by the system and sent as the delta token in the next interaction with ONEmds. The more data hint can be used by the system to decide whether it should call again immediately to fetch the next page of a bigger load or whether it should wait according to the polling interval. Events can also include the change token of the change request that triggered creation of this event.

The delta token sent by ONEmds and stored by the system is an opaque string. The format of the delta token is only known to ONEmds. The system can assume that the delta token will not contain more than 5000 characters.). Events will only include a change token in the log of the system that created the corresponding change request (positive confirmation). Create, update, and delete events will be exposed to a system or not depending on the configured instance filter for that system. Part of the master data object instances will be exposed to a system or not depending on the configured segment filter for that system

3.5.3.1 Impact for RM:

Scenario 1: Lets take a scenario where a filter has been set with cost centers A and B and initial load of replication has already happened to RM from oneMDS. Now the filter configuration is changed to add a new cost center C. According to the documentation, for cost centers A and B it should work like before with “Updated” event received for any changes, for cost center C the event “Created” should be received because this is the first upload for this cost center. So no impact in RM.

Scenario 2: Lets take a scenario where a filter has been set with cost centers A, B and C and initial and delta load of replication has already happened to RM from oneMDS. Now the filter configuration is changed to remove the cost center C or employees’s cost center changed from A to D (not RM relevant). According to the documentation, for workforce persons belonging to cost centers A and B it should work like before with “Updated” event received for any chnages, for workforce persons belonging to cost center C or for employees whose cost center changed from A to D, the event “Excluded” should be received to indicate the employees that have replicated and whose filter configurations have changed since then to have them excluded. For such excluded workforce persons the isBusinessPurposeCompleted flag at Workforce Person entity would be set to true. This assumes that DPP specific scenarios when handled the data retention manager would take care of deleting these employees and related data. Detailed impacts and handling refer table below.

Scenario 3: Now lets take a scenario where the filter C is added back into the configuration. For instances (workforce persons) corresponding to cost centers A and B, there is no change, but instances for cost center C will receive events “Included” because it is again added back, therefore appropriate action needs to be taken in RM. For included workforce persons, check if the entry is already existing in RM, and if not create the data again, if entry is existing this means that entry was previously excluded but included again before DPP relevant actions to delete the data happened. Which means “isBusinessPurposeCompleted” flag should be reset.

Summary:

Commented [PP22]: @Amalraj, Raymond -> please read these scenarios and updated table below to understand if this suffices. Have made some updates based on input from S/4 colleague on filter event handling

Commented [AR23R22]: @P, Praveen for scenario 2, instead of delimiting work assigment dates on our own, can we mark them as BLOCKED or something like "end of purpose reached ", so that those records are not used in any profile or in resoource request matching from that time onwards..In DPP scenario as well, we can take care that apart from the work assignment dates or job details dates and status, we can look at this flag as well to inform DRM that it should be deleted.

Commented [AR24R22]: @P, Praveen For scenario 3, its continuation of scenario 2. IF we use "End of purpose" flag in scenario 2 as reached, then for such records, if they are included again in OneMDS as described in scenario 3, we can remove the EoP flag so that they can put back to use. If htey have been already deleted via DRM, then we may have to create them if Work assignments are valid.

Commented [PP25R22]: @Amalraj, Raymond -> that’s a good point but on what entity do we mark this, you are talking about one flag at the workforce header entity and not at the work assignment right? But don’t we also need a date to say when this end of purpose was reached? If the work assignment end date is until end of 2021 for a delete request sent today then will you delete it only after this date?

Commented [PP26R22]: @Amalraj, Raymond -> for scenario 2 and 3 we need to decide what the system behavior should be. We can decide on flag or delimitation in date or both. For example a flag will help so that these records are not used in any transactions but if there is an assignment that is active already how does this need to be handled when a delete or exclude event is received, similarly what about the write back API in S/4. Maybe you can bring it up in the PO circle or in the next DRM discussion

Commented [PP27R22]: @Amalraj, Raymond -> I am going with a flag as per what we discussed during DRM sync when exclusion happens, please read this section and table below and get back.

Log API Event	Impact to RM	Comments
Created	Create the required data in RM	Initial Load of Employees
Updated	Update the corresponding required information in RM.	Delta Load of Employees
Deleted	No action to be taken in RM to delete the record since SFSF would only send a delete event after the retention rules in SFSF kicks in, and this information would already be updated in RM, we can not take any action for this event assuming that the DRM process would take care of the follow-on processes to delete or send the data to archival store	Follow-on DPP DRM processes should delete these employee data in RM for the data subject and legal grounds based on the DRM rules
Excluded	Data that was previously replicated is now excluded, which means such profile information for the employees are no longer valid in RM. FailureTo indicate such records for DRM process to remove, the workforce person entity has a field called "isBusinessPurposeCompleted". This flag is also required in as an indicator for inactive employees so that they are not considered in matching, assignment or any other processes	Follow-on DPP DRM processes should delete these employee data in RM for the data subject and legal grounds based on the DRM rules
Included	Data that was previously excluded is included again For included workforce persons, check if the entry is already existing in RM, and if not create the data again (SIMILAR TO CREATE event), if entry is existing previously this means that entry was previously excluded but included again before DPP relevant actions to delete the data happened.	

	Reset the “isBusinessPurposeCompleted” flag at the workforce person entity.	
REJECTED	Not sure if this applies to LOG_API because it refers to change requests and CHANGE_API	

3.5.4 *oneMDS* Multitenant Consumption

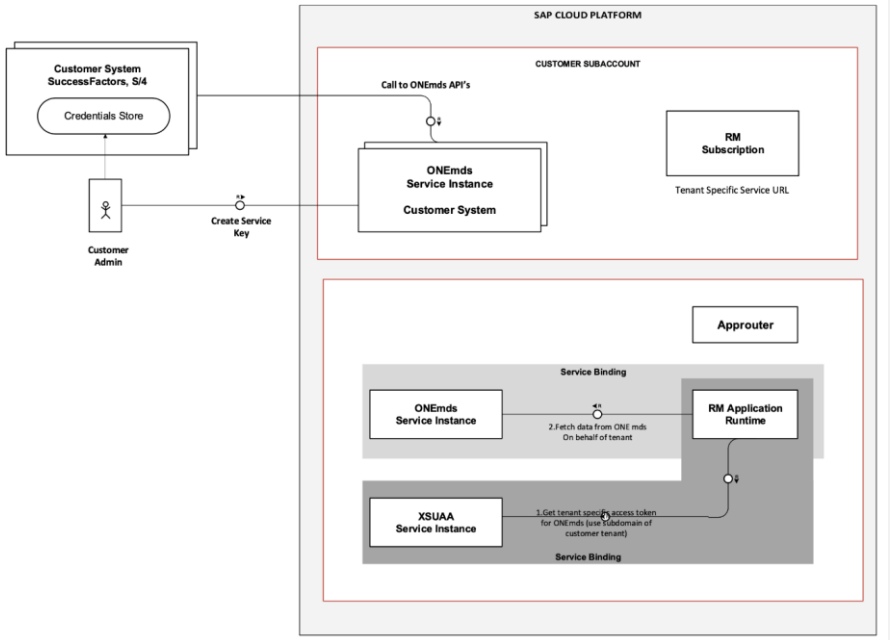


Fig: ONEmds consumption architecture approach

The approach follows standard application to re-useservice consumption using JWT token exchange flow.
Within CF bind oneMDS service instance to an application and receive the respective tenant on-and-offboarding callbacks (call-backs that a broker receives).

Get an entitlement for the oneMDS service

i. From the global account level, go to your subaccount via the tab “Subaccounts”.
ii. Click on “Configure Entitlements”, “Add Service Plans”, and select the “ONE Master Data” service while checking plan “default”.
iii. Navigate into your space and select the tab “Service” > “Service Marketplace”.
iv. Select “ONE Master Data”, select tab “Instances”, and click “New Instance”.

Bind the instance to consultant profile application as below:

a. Declare Cloud Foundry oneMDS service as ‘resource’ in “mta.yaml”.

```
- name: <oneMDS service instance name>
  type: org.cloudfoundry.managed-service
  parameters:
    service: one-mds
    service-plan: default
```

b. Bind oneMDS instance to your consultantProfile-integration-srv application as required under the "requires" section in mta.yaml.

c. Generating *oAuth* Client service token to connect multi-tenant oneMDS service.

- We be using client credentials flow with user subdomain. We need to replace the URL subdomain with the corresponding consumer subdomain for which the token is requested.
- Using the access token received, we be invoking Job Scheduler REST API to Create Job. The access token has to be used in header as Authorization: Bearer

3.6 Testability and Test Environment

- Subscription setup has to be done manually initially only.
- Destinations to be configured on subscriber sub accounts. Validate Initial and Delta Load process with multiple subscriber-based destinations.
- Integration with SFSF needs to be mocked for Pipeline and Build Tests. Since Demo system face often availability issues.

3.7 Limitations

There were few challenges as part of integration with Cloud SDK.

1. Server-side pagination not supported. Since SuccessFactors supports only v2.
   Feature Request: https://github.wdf.sap.corp/MA/sdk/issues/2865

2. Performing filters on expanded entity is not supported
   Feature Request: https://github.wdf.sap.corp/MA/sdk/issues/2914

3. Some navigations are missing in API hub due to incomplete documentation of SuccessFactors.

4. There is no provision to create tenant specific jobs from Job scheduler service Dashboard. So, to create a job specific for a SaaS tenant who has subscribed to your multitenant application, we need to exchange the JWT with XSUAA and fetch a token specific for Job Scheduler.

5. oneMDS provisioning callbacks are currently not available.

3.8 Other External Forces / Constraints and Assumptions

Following are few assumptions with which the design was considered:

1. The "*ServiceOrg* and *Costcenters*" data is available prior in the relevant tables of RM. Following is the github link: entity *costcenters*
   https://github.wdf.sap.corp/Cloud4RM/Consultant-Profile/blob/master/db/cds/core/organization.cds
2. Before initial/Delta load process start, MDO filters UI has to be set to filter employees based on cost centers.
3. Following are the filter criteria assumed for integration
   a. Filter Criteria : *isECRecord* = TRUE and
   b. Filter Criteria : *isContingentWorker* = FALSE
   c. Filter Criteria Assumption *CostCenter*
      Currently the filter criteria is to retrieve employee details based on his cost center. So, the details of cost center can be retrieved from service_org table.
4. Performance of Initial Data loading will considerably slow since the system will try to load all employee for given cost centers.
5. Fault Tolerance will be achieved with resilience4J third party library. The configurations like Timeout, Retry etc. will be decided later during the development phase.
6. We as part of integration used REST API provided by CF Destination Service for destination retrievals.
7. Availability API integration will be sub-integration within main integration. So, availability API inputs will be always fetched from database.
8. As a consumer of oneMDS service we do not own any configurations or enablement to LoB applications (SuccessFactors, S4) which could be connected to tenants.
9. Consultant profile application will bind to single service instance of oneMDS.
10. The "nextDeltaToken" attribute returned as part of response will be persisted in application and will be used for replicating delta employees.
11. The "hasMoreEvents" attribute is used to handle pagination logic.
12. The "limitHint" attribute is referred to page size indicator. If not set default 1000 records to be returned as part of LOG API.
13. The schema of the payload is validated against the schema of the ODM model. Currently this feature is not available.
14. Filter API, Instant Filters to be used replicate employees based on cost centers
15. The oneMDS tenant (DLAB/VLAB) with HR source systems will be available for end-to-end testing.
16. The service provider will provide details on multitenancy.
17. Currently the AvailabilityPeriodTypes configuration table will be filled by a CSV for GA release, future application to maintain this should be planned in future.

Commented [PP28]: Are you sure its set to 100 by default. Is this documented somewhere? We probably need to see if this is optimal and increase it to 300 if there is not so much impact.

Commented [KK29R28]: Default is set to 1000. But in our replication its set to 300.

3.9 Open Points

Commented [PP30]: Check if the open points are updated, I have updated some of them which I opened

1. How is external id mapped in S/4? - Closed
   Status: After discussion with Shreevathsa S and Karthi the agreement was in S/4 Harmonized API, wherever BP UUID is included additionally include Person Number also.
   We can relate the Person Number to the Worker.
   RM will only know the Worker and always calls EPPM Harmonized API using Worker Person Number.
   RM does not intend to integrate with BP.
2. How is external id in work assignment different from external id in workforce persons - Closed
   Status: To check with Sagar
   Update 09/03/2020: It need not be same. externalId in work assignment would be definitely different in case person has more than 1 work assignment. This field is introduced so that values can be aligned across EC and SAP ERP/S4 payroll landscape to represent 8 digit PERNR field
   EC to ONEmds integration will replicate person external id and person UUID to attribute externalId and id of WorkforcePerson in ONEmds. There is a keymapping segment in ONEmds for the assignment ids. The keymapping contains the assignment id (usersSysId) and external id in EC.

```
"systemOfRecordKeys" : [ {
    "systemOfRecordId" : "BPHIRTORETOMD",
    "isCurrentSystemOfRecord" : true,
    "systemOfRecordWorkAssignmentId" : "101010",
    "workAssignmentExternalId" : "101010"
  } ]
```

3. Emails and Phone types for replication - Closed
   Status: To check with Sagar
   Update 09/03/2020: There is a codelist (best practice). However our recommendation would be that consumer maps it to their offficial email and phone types.
   The emailType in your payload example is the internal reference number to the picklist code value. This reference number could be different in different EC systems for the same semantic meeting like business or private. EC to ONEmds integration does not send the internal reference number to ONEmds. We expand emailTypeNav and read the email type code from the field emailTypeNav/external code.
   Example:
   https://qacand-api.lab-rot.ondemand.com/odata/v2/PerPerson?$filter=(personIdExternal eq 'ESAuto22019May180946')&$expand=emailNav/emailTypeNav&$select=personIdExternal,emailNav/emailType,emailNav/emailAddress,emailNav/emailTypeNav/externalCode&fromDate=1900-01-01&$format=JSON
4. Do we need work assignment details entity
5. What field in SFSF maps to officeLocation?
6. Do we need private address mapped from SFSF?
7. What field in SFSF maps to Isbusinessprocesscompleted?
8. Do the dates map as it is from SFSF to RM or do we need some timestamp conversion to date? - Closed
   Status: To check with Sagar

Update 09/03/2020: Validity dates are floating and not timestamps. ONEmds would follow closed-open period model which is different from SF which follows closed-closed model so end date will be advanced by 1 day and would match start date of next timeslice

9. From the ID mapping table it needs to be checked how the SystemofRecordKeys entity needs to be populated and should this have a reference to the pernr for the consultant id which will be useful to send back to the WRITE APi and S/4 after assignment creation.

Status: To be discussed with S/4 integration team who are integrating with oneMDS on how this logic is and when is the SystemofRecordKeys populated.

10. From the id mapping table, its clear that WorkPerson.id in oneMDS is not generated but has a mapping to the corresponding person UUID field in Employee Central. To be discussed if we can do something similar instead of generating the GUID. Yes this Can be done - **Closed**

11. Do we need a separate configuration table for other filter criteria apart from cost centers.

12. Need to find some means to retrieve Identity Zone information for given tenant ID. - Closed

Status: No Such API available. There is development in progress for such requirement. Details on ticket:
Following is JAM question: https://jam4.sapjam.com/questions/ylt6DVbLLdmowR47CMueML
They are building an API for this purpose, when this would be available, we should stop persisting, that's the directive:
https://jtrack.wdf.sap.corp/browse/CPCORE-2275

14. Find out about Job scheduler multitenancy and how we can get the tenant specific details (identity zone) during the job run. - Closed
    - Either we keep a table persisted with this information (cross tenant) during onboarding or figure out how to derive this through any available API

Status: The other possible way to retrieve *uaaSubDomain* details for all subscribed tenants. Basically, the SaaS Instance manager API, *Get All Subscription* API returns URL which contains uaaSubdomain.so we can extract the subdomain out of it. With this we don't have to store tenant details in cross tenant storage

https://wiki.wdf.sap.corp/wiki/display/CPC15N/SaaS+Application+Registration+in+CF#SaaSApplicationRegistrationinCF-Getapplication'ssubscriptions

15. How Data Filtering API registration process will be automated for a system of a tenant? Since for canary we need to send request to oneMDS team for approval.
16. Need some inputs on excluded or included events after filter engine is activated? - Closed
18. How can we activate multiple filters? Since registering new filter will deactivate previous filter criteria.
19. How to fetch *serviceInstanceID* required for Filtering API from service keys
20. When will be the service broker callbacks available as part of *oneMDS* Multitenancy?
21. How fast the data is replicated from HR source systems to oneMDS? Accordingly, we can configure schedule time for delta employees' replications.

22. Service Org in Organization and details table: How are we planning to populate data into these tables. If this is an APP, have we decided who is going to do this and when?
    Status: This will be an APP until an interface view from S/4 is ready.
23. Understanding of the typical data volume we are dealing with here.
    a. Profile data -> how many cost centers would typically be involved for Resource Management and how many employees per cost center on an average
    b. Availability -> how many records for 2 years of data considering the time period from start of year of replication to end of next year for one employee.
    c. How often on an average does profile data relevant in RM change for an employee
    d. How often on an average does availability data change for an employee
    e. Extrapolating results from a,b,c and d will give an idea of the data volume we are dealing with for both initial and delta load
24. The oneMDS filter concept is undergoing a change and the new documentation and information around how filters work is still being explored
25. The filters required from RM is still not supported by oneMDS, point 27 needs to be clarified to understand how filters work and a proposal from RM on what filters are needed should be sent to the oneMDS team.
26. Clearing records failed during the availability upload failures for keys like start date, resourceId and cost center need to decide.

3.10 Complex Algorithms and Applied Patterns

NA

3.11 Design Alternatives and Trade-Offs

3.12 Guide to the Implementation

To be updated

4 Appendix

4.1 Glossary

Term	Abbreviation	Definition
Cloud Application Programming	CAP	Programming model to be used to implement application on SAP Cloud Platform
Resource Management	RM	Resource Management is the name of the project under the Cloud for Projects portfolio
SAP Cloud Platform	SCP	SAP's PaaS platform to host cloud applications
Fiori Elements	FE	SAP's UI Development framework Used to create frontends based on annotations
SFSF	SuccessFactors	SuccessFactors is a suite of SAP products to provide a cloud-based solution to manage diverse HR functions

Term	Abbreviation	Definition
EC	Employee Central	EC in SFSF consists of employee, foundation and metadata framework (MDF)_objects.
DAO	Data Access Object	Data Access Object Pattern or DAO is design pattern which is used to separate low level data accessing API or operations from high level business services
	Command Design pattern	Design pattern under behavioral pattern category. A request is wrapped under an object as command and passed to invoker object. Invoker object looks for the appropriate object which can handle this command and passes the command to the corresponding object which executes the command
oneMDS	One Master Data Service	ONEmds for short) will serve as central entry point for *master data integration* within the *Intelligent Suite*.

4.2 Customizing

Updated as part of section 3.2.

4.3 Supportability Considerations

Not Applicable

4.4 Error Analysis

4.4.1 Debugging

To be updated

4.4.2 Logging and Tracing

Application Logging Service available as part of the standard Cloud Foundry offering will be used for standard logging and tracing.

1. The logger level at the root is warning and debug as logger level for com.sap.c4p.rm.consultantProfile package.
2. All the successful logs are logged as debug logs and all error scenarios are logged using error logs.
3. Correlation ID is used to track for all replication-based error log details in ELK stack.

4.4.3 Other Error Analysis Tools

Not Applicable

4.5 Other

Not Applicable