|  |
| --- |
| it.archive  Technical Design Document |

Table of contents

[1. Introduction 4](#_Toc76639369)

[2. General Architecture 4](#_Toc76639370)

[2.1 Solution Architecture 4](#_Toc76639371)

[2.2 Technical architecture 5](#_Toc76639372)

[3. Infrastructure 6](#_Toc76639373)

[3.1 it.archive client 6](#_Toc76639374)

[3.2 it.archive backend 6](#_Toc76639375)

[3.3 Yuuvis Momentum backend & Keycloak 6](#_Toc76639376)

[3.4 SuccessFactors 6](#_Toc76639377)

[3.5 Customer’s identity manager 7](#_Toc76639378)

[4. Authentication & Authorization 7](#_Toc76639379)

[4.1 Authentication & SSO 7](#_Toc76639380)

[4.2 Authorization & User management 7](#_Toc76639381)

[4.3 Roles 8](#_Toc76639382)

[4.4 Organizational relationships 9](#_Toc76639383)

[5. User client UI 9](#_Toc76639384)

[5.1 Employee-role dashboard 10](#_Toc76639385)

[5.2 Dashboard for Manager & Content admin 10](#_Toc76639386)

[5.3 Solution administrator 11](#_Toc76639387)

[5.3.1 Solution Administration UI 12](#_Toc76639388)

[5.4 Document creation UI 12](#_Toc76639389)

[5.4.1 Manual Tagging 13](#_Toc76639390)

[5.5 Manual document deletion 14](#_Toc76639391)

[5.6 Document viewing/rendering 14](#_Toc76639392)

[5.6.1 Direct viewing 14](#_Toc76639393)

[5.6.2 Supported document types 14](#_Toc76639394)

[5.7 Other UIs 15](#_Toc76639395)

[5.8 Sensitive Data report 15](#_Toc76639396)

[5.9 Purging report / UI 15](#_Toc76639397)

[6. Automatic document maintenance 16](#_Toc76639398)

[6.1 Employee master data changes in SF 16](#_Toc76639399)

[6.2 Purging 16](#_Toc76639400)

[6.3 Document archiving from SuccessFactors 17](#_Toc76639401)

[6.4 Document archiving from it.capture 17](#_Toc76639402)

[6.4.1 API definition 18](#_Toc76639403)

[6.5 Automatic tagging 18](#_Toc76639404)

[6.5.1 via it.capture 18](#_Toc76639405)

[6.5.2 via SuccessFactors 19](#_Toc76639406)

[7. Audit trail 19](#_Toc76639407)

[8. Languages 19](#_Toc76639408)

[9. Artefacts and data model 20](#_Toc76639409)

[9.1 Document types 20](#_Toc76639410)

[9.2 Documents 20](#_Toc76639411)

[9.3 System configuration 21](#_Toc76639412)

[9.4 Tenant list 21](#_Toc76639413)

[9.5 Tenant-specific settings 21](#_Toc76639414)

[9.6 User & employee data 21](#_Toc76639415)

[10. Operation 22](#_Toc76639416)

[10.1 Tenant management 22](#_Toc76639417)

[10.1.1 New customer 22](#_Toc76639418)

[10.1.2 Customer leaving 22](#_Toc76639419)

[10.1.3 Changes to tenant 22](#_Toc76639420)

[10.2 Usage limitations, consumption count & invoicing 22](#_Toc76639421)

[10.3 Backup strategy 23](#_Toc76639422)

[10.4 Availability/uptime and fallbacks 23](#_Toc76639423)

[10.5 Performance and scaling 23](#_Toc76639424)

# Introduction

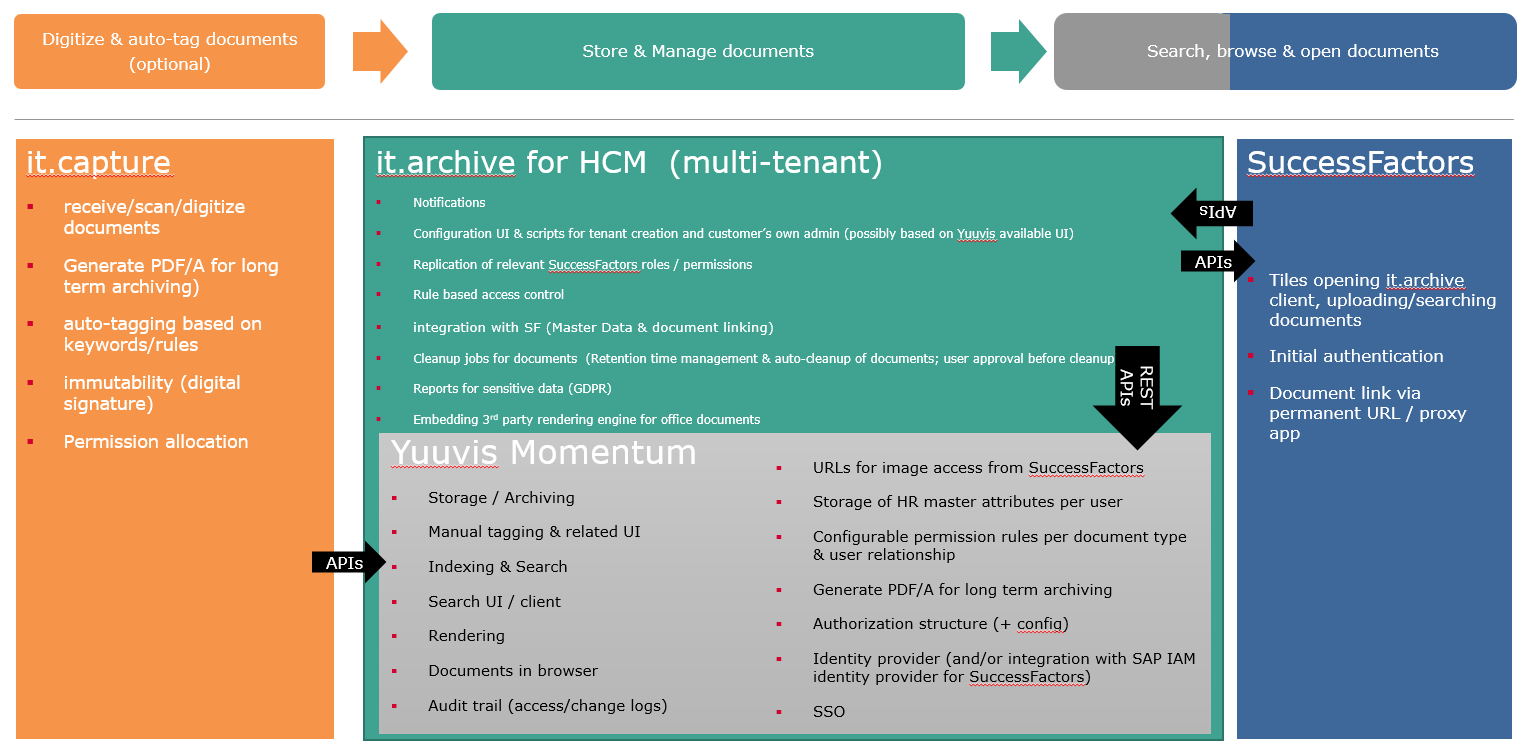
Below technical design describes how the *it.archive for SuccessFactors* will be implemented to fulfil the functional requirements outlined in the HR Archive – Functional Description v1.0.

It aims to define the solution architecture, application components and their main functionality, technologies used and related aspects (like security, hosting etc.)

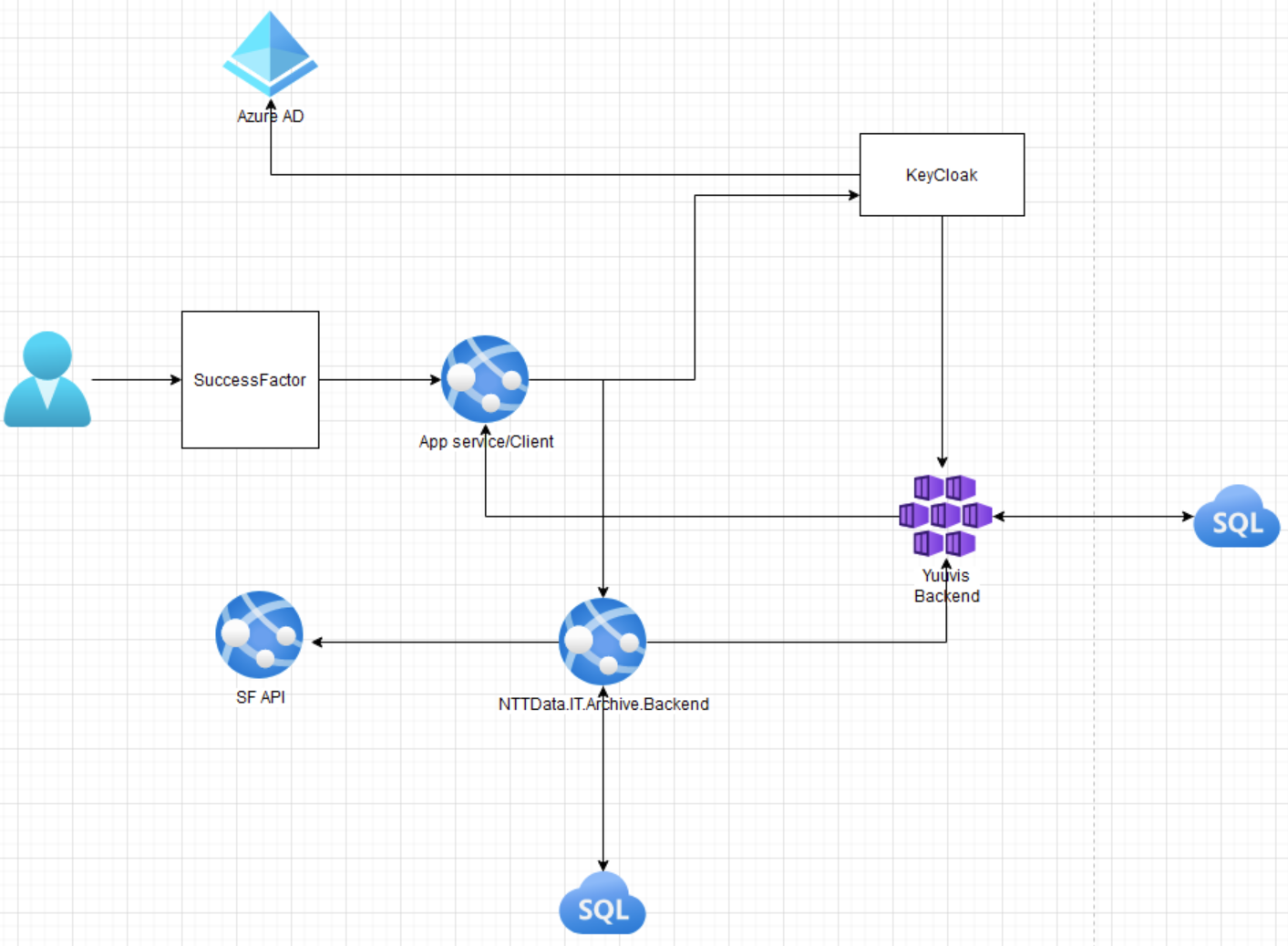
Requirements are referenced via their section numbers from the functional specification document.

# General Architecture

## Solution Architecture



## Technical architecture



it.archive is an independent application for document storage & management that seamlessly integrates with SAP SuccessFactors using single sign-on (and is open for integration with other systems). It comes with it’s own client, adjusted and branded to provide a fairly integrated look and feel with SF, and it’s own backend to implement custom features.

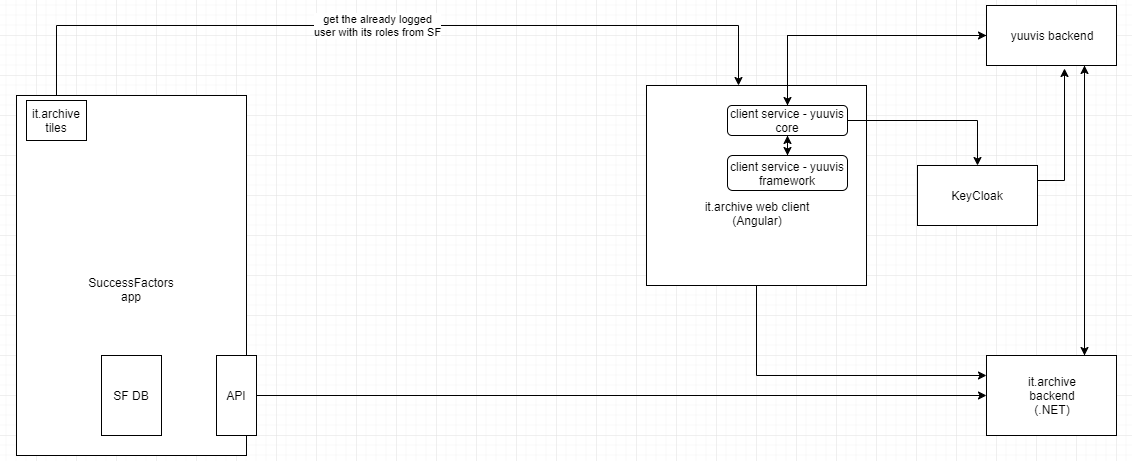
it.archive will be a scalable multi-tenant solution (multiple SuccessFactors customers served by the same application). In the most common deployment scenario NTT Data will be hosting a 2 tenants (dev/test & prod) per customer.

There will be one client (User-Interface) made with Angular 11.0 and one backend service made in DotNet xxxxx.

The client will connect to Yuuvis using their existing Angular components to utilize their implemented functionality, however the client will also connect to the it.archive DotNet backend to utilize custom implementations of it.archive-specific functionality and to request data from SucessFactors.

The Keycloak IDP hosted alongside the Yuuvis platform in the Yuuvis.com domain will be used to integrate SuccessFactor IDP with Yuuvis authentication and it.archive authentication to fulfil the SSO requirement.

Roles in SuccessFactors will be mapped to it.archive roles.



# Infrastructure

## it.archive client

A public Angular 11 client, hosted in 1 Azure app service managed by NTT Data.

Significant reuse of Yuuvis reference client code.

Connections from the user to the it.archive client will take place over the internet encrypted with SSL/HTTPS.

## it.archive backend

it.archive backend will be hosted in 1 Azure app service managed by NTT Data.

Accessible from the internet on it.archive domain (tbd)

## Yuuvis Momentum backend & Keycloak

Optimal Systems GmbH hosts and manages the *Yuuvis Momentum* instance(s) (currently a shared cluster) providing document management and document storage features, as well as *Keycloak* for identity management/integration.

Connections to Yuuvis API will take place over the internet encrypted with SSL/HTTPS.

## SuccessFactors

Connection from SuccessFactors to it.archive will be done via static URLs (tiles) to different UI components of it.archive. Information about the currently logged on user will flow from the customers IDP.

Connection from it.archive to SuccessFactors and each customer’s instance/tenant, will be done via whitelisted APIs only. A technical SAP SF user with RBP authorization will be needed for it.archive in order to obtain data from SF.

Connections to SF will take place over the internet encrypted with SSL/HTTPS.

## Customer’s identity manager

OpenID Connect compatible IDPs like SuccessFactors IDP will be used for authentication. These are hosted by the customer, SAP or other 3rd parties.

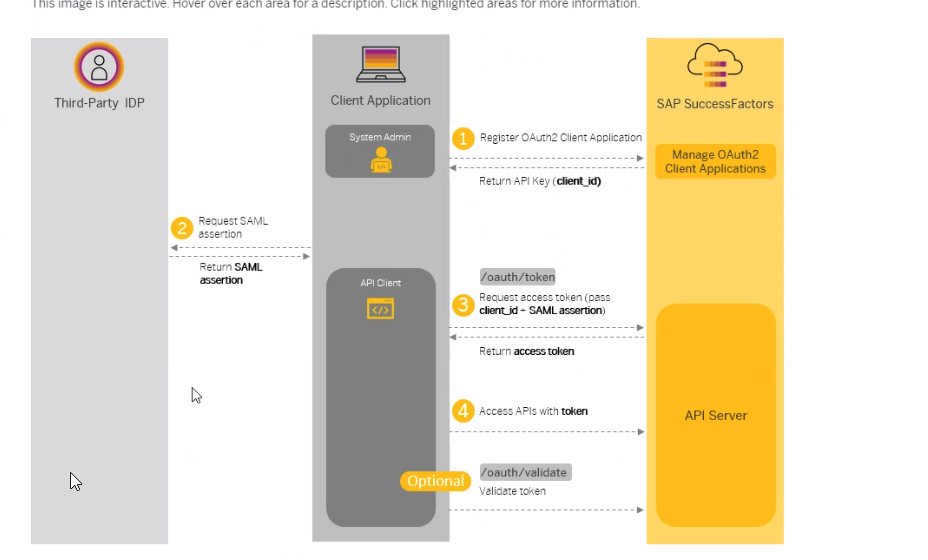
# Authentication & Authorization

## Authentication & SSO

Functional requirements: 2.4

it.archive will support OpenID Connect compatible IDPs by configuring trust in it.archive/Yuuvis Keycloak for these IDPs.

As the it.archive client will be started from the SF application, no additional login form is required and single sign on (SSO) using KeyCloak identity provider will be used, providing seamless authentication of already logged users in SF application



## Authorization & User management

Functional requirements: 2.4, 2.5, 4.2

* Users will be managed in SuccessFactors.
* SF IDP will be trusted by it.archive & Yuuvis (see authentication section).
* User data will be replicated to the Yuuvis Keycloak for authorization purposes using LDAP which will satisfy the requirements of handling expatriated employees
* SF user roles will not be synchronized to it.archive, but will be retrieved from SF when needed using Yuuvis backend webhook *<UserInfo>*. The webhook will allocate Yuuvis permissions (as configured for it.archive roles) to the logged on user for the active session. The webhook will be configured in each Yuuvis tenant, calling the web service xxxxx exposed by the it.archive backend, which in turn connects to customer’s SF instance using the whitelisted API xxxx SF via a technical user (authorized to query the whole user data).
* technical SF user credentials will be stored securely in Azure keyvault (multi-tenant)
* Login screen appears if user was not signed on or SSO wasn’t possible.
* Local it.archive/Yuuvis (non-SF) users can be defined in keycloak (e.g. NTT DATA admin users), but this is not the normal flow for it.archive for SF and not a core requirement for phase 1.

## Roles

Functional requirements: 2.5

* 4 it.archive roles will be available (multi-tenant it.archive SQL table)
  + Employee
  + Manager
  + Content admin
  + Solution admin
* it.archive roles get allocated Yuuvis permissions (multi-tenant it.archive SQL table).
  + Maintained during tenant creation (by script by NTT Data)
* mapping of SF roles to it.archive roles will be stored in the it.archive SQL db

|  |  |  |  |
| --- | --- | --- | --- |
| Tenant/customer ID | it.archive Role | SF Role name | Yuuvis permissions |
|  | Employee | Employee View of Self | Tbd |
|  | Manager | Manager | Tbd |
|  | Content Administrator | HR Partner | Tbd |
|  | Solution Administrator | Archive Administrator | Tbd |

* SF OData API
  + Description from Michael/Mikael
* it.archive roles grant Yuuvis permissions (Yuuvis webhook handles that)
  + Alternatives: Composite roles based on standard yuuvis roles or modifing yuvis roles to reflect the logic like e.g. employee can create documents only for him/herself. Manager can create documents only for his employees. (Manager users also have Employee role so they can create documents for themselves) …
* Yuuvis permissions will be modelled to use “employee ID” as criterion to grant access to documents. it.archive will dynamically grant permissions create, read, update or delete to a user for documents of certain “employee IDs”

## Employee Relationships (SF Standard relationships)

Functional requirements: 2.5

* Manager -> Employee relationship is available in SF
* it will be evaluated to grant a manager user access to employees data reporting directly to them (or indirectly, manager of manager or content administrator)
* Relationships will not be replicated but read on demand from SF using API xxxx in order to avoid data storage, management, ownership questions and replication delays.
  + Webhook retrieves all employee IDs during logon from SF that they have access to (manager, content admin=HR BP, indirect manager)
  + Relationship changes will be reflected when the user signs in after logout or expiration of the session

# User client UI

Functional requirement: 2.13, 2.11, 2.12, 3.3, 2.3

Standard Yuuvis UI components will be used for all client functionality unless otherwise specified. Permission based access to features will be implemented by default configuration for the 4 it.archive roles. This is delivered out of the box, should normally not be changed during implementation projects in order to preserve the integrity of the roles, but can be adjusted by NTT Data, if agreed.

Yuuvis standard Angular 11 reference client source code is imported into the it.archive client project and reused. Configuration, stylesheets and static page layout is used to achieve the it.archive look and feel.

Code adjustments to Yuuvis client components will be made after careful consideration only where really necessary in order to preserve the possibility to upgrade to newer versions of the reference client.

All it.archive roles will have a main menu, role-dependent dashboard, search, role-dependent features for document creation and user/client settings.

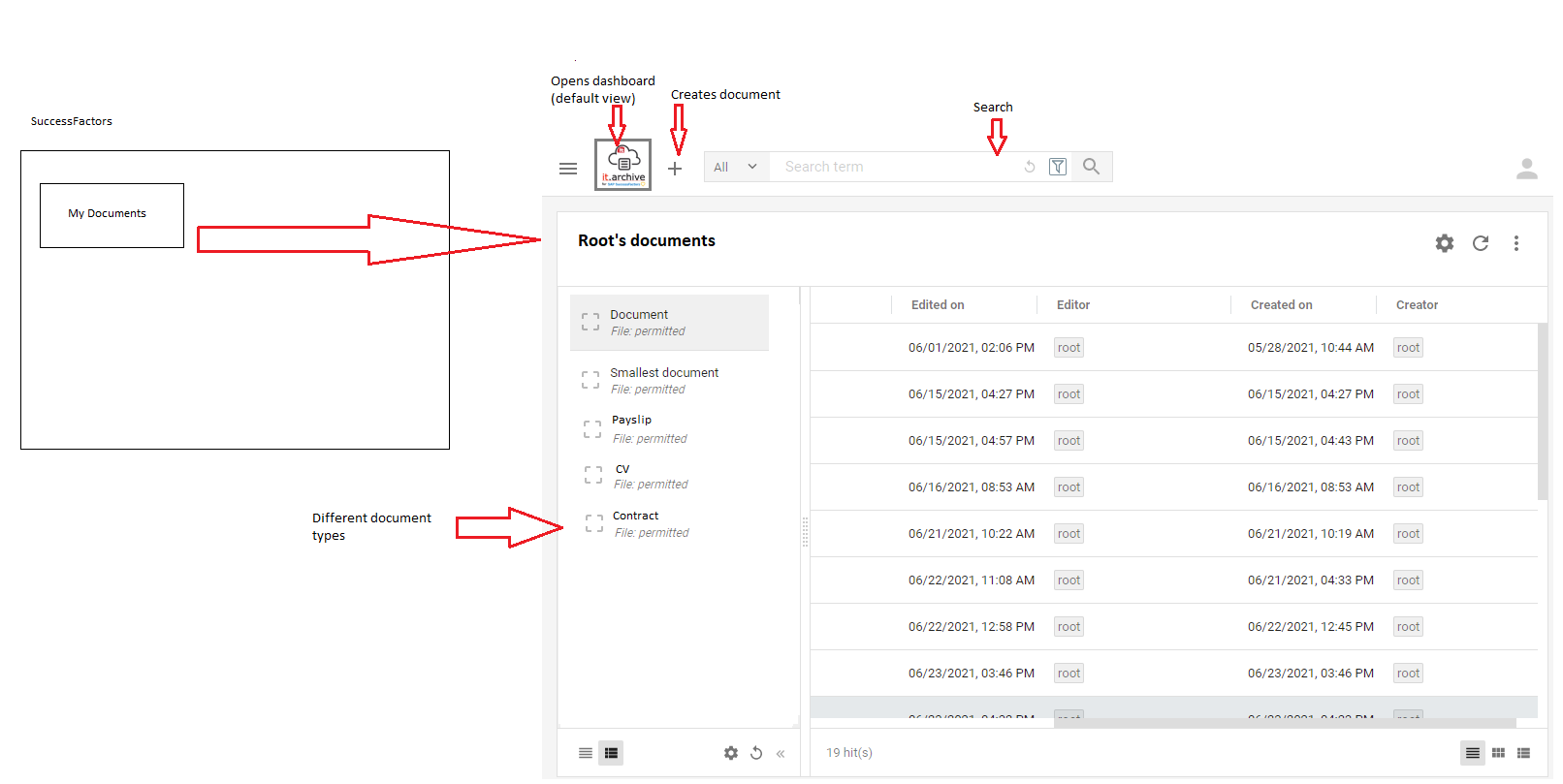
The role-specific default dashboard is opened when starting the client without any specific target URL.

Other views can be opened by static URLs (e.g. all documents with document type xyz). :

The relevant SF tiles will thus have to be configured during the implementation project with possible customer specific document type IDs.

## Employee-role dashboard

In the default view, an employee will see all their documents grouped/filterable by document type



Audit trail is excluded by means of Yuuvis permission.

## Dashboard for Manager & Content admin

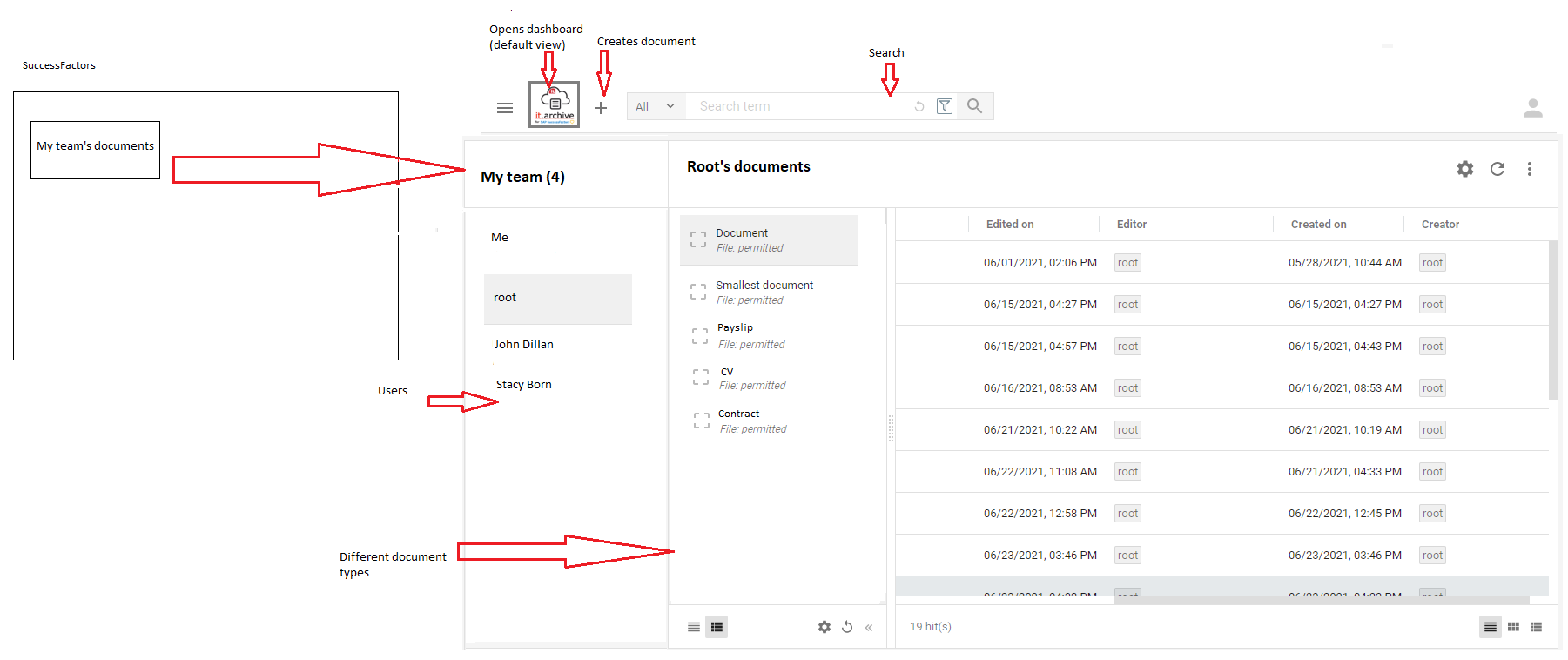
In the leftmost pane, the default dashboard for Manager and Content admin shows a list of employees (Name & ID) that they can manage. This list is retrieved via it.archive backend from SF during user logon to it.archive client. Search possibility in employee list for content administrator.

Selecting a user from the list of employees, shows their document types in the second pane and the employee’s documents in the 3rd pane. Count of individual employees will be shown.

Selecting a document collapses the 2 leftmost panes and opens the document for viewing in the 4th pane (standard Yuuvis object viewer giving permission-dependent access to metadata, audit trail and rendering).

List of the document types the selected user has documents for can be filtered.

Documents can in addition be filtered by basic metadata.



For Manager role the Audit trail is excluded by means of Yuuvis permission, but for Content Administrator it is available.

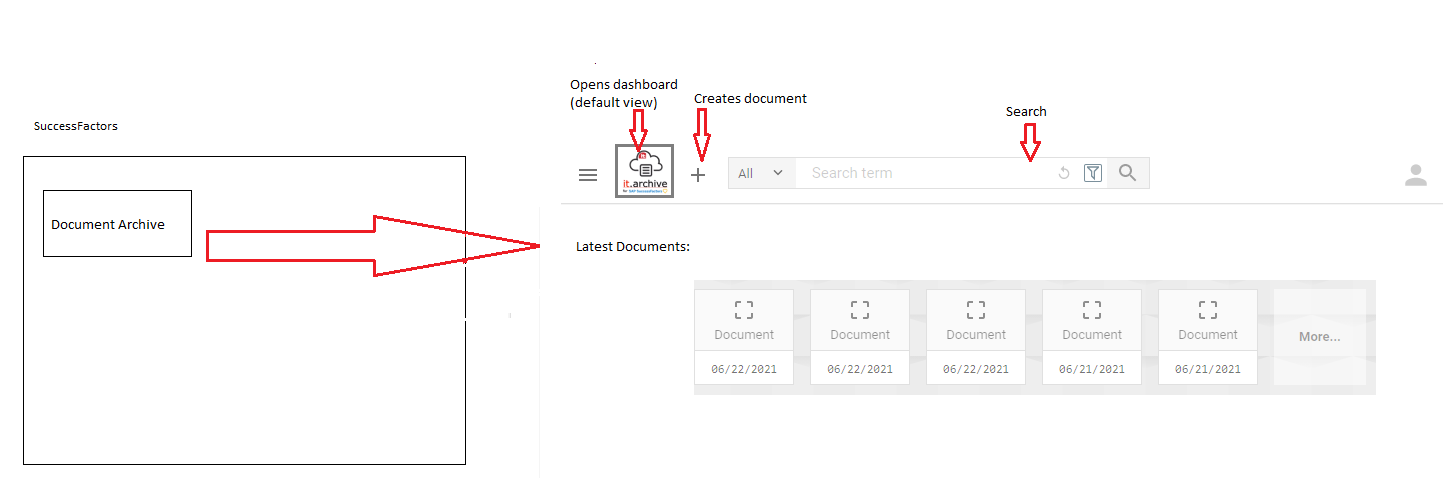
Managers & Content administrators can also open the Employee dashboard to see their own documents.

Content Administrator’s access to employees deleted in SF via SF relationship or via location (tbd).

## Solution administrator

No document dashboard is shown to Solution Administrator.

Recently used document will be shown. For example: show latest document of specified type (configured in the tile -> type= Document)



As Solution Administrator role has no restrictions on document type, employees, locations etc., all employees & documents are shown when opening the Manager/Content admin UI (“My Team”)

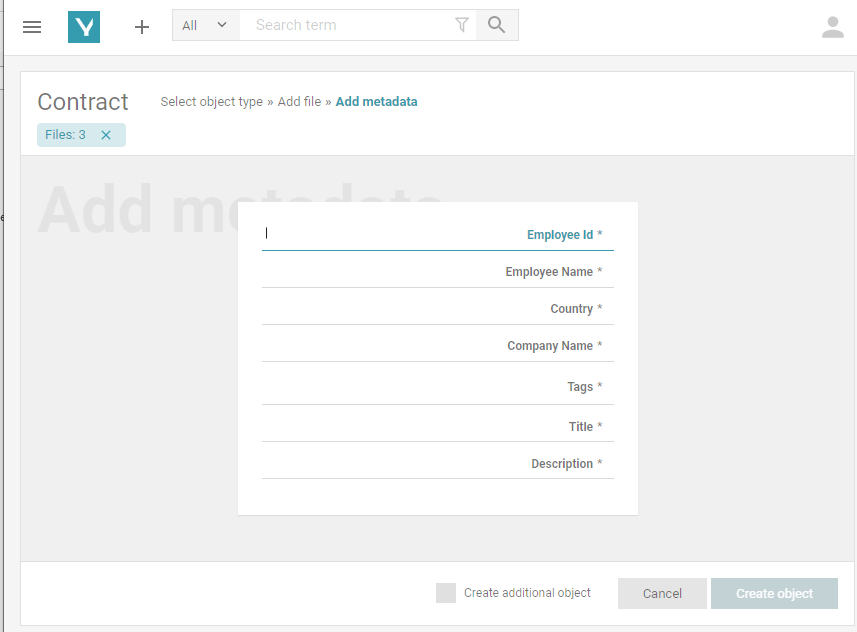
### Solution Administration UI

* Accessible via the menu to users with Solution Admin role
* Specific Angular component development for it.archive and integrated with Yuuvis reference client & it.archive backend
* Allows tenant specific changes/actions as follows:
  + (phase 2) Maintain purge rules
    - Retrieve, view, edit & save and delete all purging rules (via API calls to it.archive backend for purging rules and SF data and Yuuvis backend for document types)
    - List of all purge rules in the tenant with name and properties (from it.archive db table)
    - Editor UI for changing the purge rules, deleting, creating new rules
    - No checks for duplicates or consistencies
  + (phase 2) Maintain document types & tags
    - Retrieve, view, edit and save all document types (via API calls to Yuuvis backend)
    - Edit name and field
    - Mark document type inactive (= property on document type)
    - Maintain a list of tags per document type & tag field (= field properties in Yuuvis schema)
      * Create new
      * Remove (tag remains on already tagged documents, but cannot be allocated new)
    - (phase 2) Edit field allocation, field properties (Yuuvis schema)
    - (phase 2) create new document types
    - (phase 2) Delete document type if no documents are existing (Yuuvis ensures this)
  + Role management
    - (Phase 1) NTT Data configures changes to the solution on request by maintaining SQL table directly
    - (Phase 2) UI allows to edit it.archive roles
      * Names
      * Yuuvis Permissions
      * SF roles mapping to it.archive roles

## Document upload UI

Functional requirements: 2.3

* One or multiple documents of one doc type can be uploaded, then employee is selected and metadata is fetched and defaulted
* Documents can be uploaded manually via it.archive client (“+ button”) and selection or Drag/Drop of one or multiple file(s)
* User cannot change location, employee name & company name. This data can only come from SuccessFactors.



* Based on upload component of Yuuvis reference client
* Extended with dropdown field for Employee ID and Employee Name, which are populated with employees that the user has access for (held for the session lifetime)
* Other fields are defaulted when user is selected (defaults are retrieved from SF by means of an API call to it.archive which in turn calls SF API for user master data)
* User with Employee role gets their employee ID pre-defaulted
* Manager, Content Admin roles get a selection of employee IDs that they have permission for
* it.archive client calls it.archive backend which calls SF API in order to prefill employee master data
* Fields/properties
  + Title (defaulted from file name)
  + …

### Manual Tagging

Functional requirements: 2.12, 2.1.4.2

(Phase2 )

* Manual tagging is possible during document creation and editing (requires metadata Edit permissions)
* 1 multiple value “Tags” field will be included in each document type
* Dropdown for list of possible tags
* Custom tag field is configurable per document type *(see Solution Administration UI)*
* UI for maintaining tag list per custom tag field *(see Solution Administration UI)*
* Tag removed from list will not be available for new use anymore, but remains on documents
* Searching by tags supported

## Manual document deletion

No separate UI for normal document deletion is created.

Users with delete permissions for the relevant documents can manually delete individual documents in their normal views.

TBD with Yuuvis if multi-file selection for deletion can be enabled in normal standard views.

Manual document deletion should only mark documents for deletion. Default purging rule will be defined for documents marked for deletion by users. Purging agent will then notify content administrator and propose for permanent deletion.

*(For purging see separate section.)*

## Document viewing/rendering

Functional requirements: 2.9; 4.3.1

### Direct viewing

The standard Yuuvis document viewer component will be used.

It will be extended so it can be opened with the most recent document of a particular type using a static UI with URL parameters:

* “doctype”=… (Yuuvis document type)
* “latest”=yes

This will open the viewer showing the document of the particular type with the most recent creation date.

*e.g.* [*https://nttdata\_itarchive.com/client/viewer?latest=yes&doctype=employementcontract*](https://nttdata_itarchive.com/client/viewer?latest=yes&doctype=employementcontract)

### Supported document types

All document types can be stored.

List of document types supported by Yuuvis for rendition (PDF conversion and/or viewing directly):



Besides the document types supported natively by the document viewer, Yuuvis Rendition Services will be configured to render PDF versions of Office documents during upload, in order to allow showing the documents in the native viewer without downloading to the users PC.

[*https://help.optimal-systems.com/yuuvis\_develop/display/YMY/Rendition+Endpoints*](https://help.optimal-systems.com/yuuvis_develop/display/YMY/Rendition*Endpoints)

For Excel documents a 3rd party viewer will be embedded in the Yuuvis viewer, by using the standard extension mechanism. (component TBD).

*[https://help.optimal-systems.com/yuuvis\_develop/display/YMY/Viewer+Service](https://help.optimal-systems.com/yuuvis_develop/display/YMY/Viewer*Service)*

## Other UIs

Standard Yuuvis components will be used for all other UI elements.

Document editing will not be supported.

Document versioning will not be supported.

## Sensitive Data report

Functional requirements: 2.1.2, 2.1.3

A report will be available to Employee, Manager & Content Admin (for employees & document types they have access to) & Solution Administrator role (for all users), showing a list of documents filterable by sensitive data flag. The report contains:

* Employee number
* employee name
* Document Type
* Document name
* document date
* Expiration date (proposed auto-purging date)
* ”Sensitive data” flag
* ... other basic metadata

Phase 2: CSV export

## Purging report / UI

* + Accessible only by solution admin role
  + The report lists all documents that are candidate for purging
    - requests purging rules via API call to it.archive backend
    - dynamically generates relevant queries to Yuuvis backend
    - executes queries
    - retrieves matching file metadata
  + Report shows at least:
    - Employee ID
    - Employee name
    - Document type
    - Creation date
    - Original purging/expiration date (calculated property based on purging rule)
    - Extended purging date
    - Applicable purging rule
  + Content filters & sorting are available
  + Multi-selection supported
  + Button “Delete”
    - Shows warning: Do you really want to continue?
    - Performs deletion of documents (with the permissions of the executing role)
  + Button “Extend retention”
    - Updates the “Extended on date” property (=date on which the retention period been extended) on the documents, based on the definition of allowed extension period in purging rule (“Extended on” date + x = extended purging date)

# Automatic document maintenance

## Employee master data changes in SF

Functional requirements: 2.7, 2.8

* We will not replicate employee master data from SF, but only retrieve it in order to populate defined document attributes when a document is created, or update these attributes when SF master data changes
* Agent (Task service) will refresh document metadata:
  + Employee name master data (not document master data)
* Either triggered on nightly basis, or event from SF (preferred; tbd if possible)

## Purging

Functional requirements: 2.11

* Purging rules are stored in an it.archive multi-tenant SQL db table
* (phase 1) No UI for purging rules.
  + it.archive will come with predefined/proposed rules adhering to current legislation (NTT HCM to configure these).
  + Rules are populated during tenant creation and can be adjusted to customer needs by NTT Data during implementation project or later on demand.
* Agent (Task service) will
  + search expired documents
  + notify content admin in notification bar (Yuuvis inbox)
* Custom tag fields (with value list) will be possible to use in purge rules
* *Status* will be calculated on-the fly based on termination date
* UI for reporting & approving *(see UI section)*

## Document archiving from SuccessFactors

Functional requirements: 4.1

Relevant for phase 1

* + Recruiting/hire process - Recruitment: Cover letter, CV, Other documents

Phase 2

* + Onboarding: Access requests, PC ordering, Other documents
  + Compensation: Compensation statement
  + Variable Pay: Bonus statement
  + Platform Document Portlet: misc. HR documents
  + Employee Central: workflow attachments, e.g., the maternity certificate
  + Third-party pay engine: Pay slips

An agent in it.archive that:

* gets triggered by an event sent from SF via it.archive API
  + Event tbd
  + API tbd
  + Document types/processes tbd
* looks for new documents in SuccessFactors
* Imports the documents to it.archive
  + Download document
  + Collect process/document type specific metadata (this does not support custom metadata, but predefined, hardcoded metadata per process)
* Deletion

## Document archiving from it.capture

Functional requirement: 2.3.2; 3.1

Covered in phase 1 is the technical it.capture connector for it.archive, supporting

* Yuuvis document upload API
* it.archive API for SF employee ID search

Business requirements need to be determined in more detail to be able to define the profiles and offer a value added functionality.

General flow:

1. Document sent to dedicated email operated by the customer, monitored by it.capture
2. Data extraction
3. Document type detection from email subject, file name, sender and/or document key words
4. Route to job/profile for customer/tenant and
5. Determine data
   1. Determine employee name, employee ID if available, language of the document, country from address
6. Image archiving in it.archive via REST API exposed by Yuuvis

### API definition

(see Yuuvis documentation)

* SSL/HTTPS
* Technical it.capture user / API key per tenant
* Identifier search API calls it.archive backend API to determine employee ID from
  + Employee name
  + National register number
* Upload API requires the mandatory data defined per document type
  + Phase 1: only core document fields (available for all document types) are supported
    - Employee ID
    - Employee Name
  + (Phase 2) additional document type specific fields will be supported
    - Profile fields, mapping & value determination logic to be described/pre-defined in detail for supported document types
    - mapping of profile field to API parameter is configurable in it.capture

## Automatic tagging

Functional requirements: 2.12

The requirement is currently not solved and needs to be clarified.

Where would the information come from for automatic tagging?

### via it.capture

General flow see above.

In addition there will be tag determination based on key word search

Supported document types & tag determination logic to be defined

### via SuccessFactors

(Phase2)

Supported document types & tag determination logic to be defined

General flow:

Static link URL parameter for tags allows to prefill multi-value document property <tags> with values

<http://server/create?tag=a&tag=b>

tbd: Custom fields to be supported during document creation? FuncSpec 4.1?

We do not expect to support coding custom logic on it.archive side for each customer implementation. While this is technically possible, it is not part of the current concept.

# Audit trail

Functional requirements: 2.10

Document history is available for specific documents using standard Yuuvis UI components. Filtering is supported.

Read, write, etc. ….whats in the standard

Phase2:

The combined audit logs for multiple documents can be downloaded as a flat list to CSV file via a menu entry in the *Sensitive Data* report. Filtering is observed, so the audit log export comprises of all documents that are shown in the report/list after filtering. Audit trail permissions are required to have the “Export” menu entry available. Document type permissions, employee permissions and audit log permissions will be observed when generating the export.

Audit logs will expire and be purged after specified time (based on document type configuration) without further visualization in the it.archive client.

Audit logs do not exist independent of documents (purging a document purges the audit logs).

# Languages

Functional requirements: 3.3

it.archive client (& backend messages) will be available in English.

German translation will also be available.

Other languages can be added by providing a json file with needed translation strings.

# Artefacts and data model

## Document types

* it.archive document type = Yuuvis object type. Each object type has a set of “metadata properties/yuuvis tags” that apply to all documents of the type.
* Object types that will be supported may include:
  + Recruitment: Cover letter, CV, Other documents
  + Onboarding: Access requests, PC ordering, Other documents
  + Compensation: Compensation statement
  + Variable Pay: Bonus statement
  + Platform Document Portlet: misc. HR documents
  + Employee Central: workflow attachments, e.g., the maternity certificate
  + Third-party pay engine: Pay slips
* Core properties of each type are:
  + *Sensitive Data* flag (“yes/no”; documents of this type can/do contain sensitive data).
    - Not editable on documents
  + ….tbd Additional required properties of the above-mentioned document list should be further discussed.
* Document types will be maintained by NTT Data during tenant setup
* Additional document type properties will be modeled during tenant set-up.
* If purging for specific document type must be maintained, this information should be preserved in dedicated table in database. Aim is to store this in the Yuuvis backend, but alternative solution is a dedicated table in the multi-tenant it.archive db.

## Documents

* Each document has one document type
* Each document type has properties that define it and are inherited by the document
* Fields/metadata per document are defined in the Yuuvis schema per document/object type
  + All documents
    - Employee ID (= SF User ID)
    - Employee Name (this is updated when changed)
    - Employee location (=country the employee is currently employed in; this is updated when changed)
    - Legal entity of employee (=company the employee is employed by; this is updated when changed)
    - Document Type
    - Termination date
  + Document type-specific fields are possible

## System configuration

* Yuuvis API configuration stored in Azure key vault
* Credentials / API key for it.capture API access generated and managed in Azure API management & key vault
* …

## Tenant list

* Tenant IDs & customer names stored in Keycloak (to be verified)
* Users IDs per tenant are stored in key cloak
* definition of tenant via URL parameter (e.g. from SF)
* it.archive SQL table storing it.archive tenant ID & Yuuvis tenant ID
  + customer name
  + it.archive unique tenant ID
  + Yuuvis tenant ID

## Tenant-specific settings

* it.archive SQL table containing
  + it.archive roles with their mapping to Yuuvis permissions (1:n)
* it.archive SQL table containing
  + it.archive roles with mapping to SF roles (n:n)
* SF credentials for technical API user stored in key vault

## User & employee data

* User IDs stored in Keycloak & SF
* Employee & user data stored only in SF
  + (Phase 2) caching in it.archive if required for performance reasons
* Employee data for reporting / document retrieval stored on document metadata

# Operation

The solution will be operated by NTT Data resources (AMS? tbd)

## Tenant management

* Tenant metadata will be stored in a it.archive for SF SQL table (multi-tenant)
* maintenance will be manual in the first version (no it.archive system admin UI)

### New customer

* + Tenant creation (yuuvis & it.archive)
  + Yuuvis technical user & API key setup
  + Tenant metadata
  + Yuuvis Webhook setup for it.archive integration
  + it.archive configuration for SF integration (endpoints & credentials)
  + Setup user data replication from customer’s IDP (via LDAP)
  + Keycloak trust config for SSO
  + Document type adjustments (e.g. what is sensitive data, custom document types)
  + Document purging agent setup
  + Audit trail purging setup
  + TBD: Initial data migration via RPA?

### Customer leaving

* + tenant deletion Yuuvis & it.archive

### Changes to tenant

tbd

## Solution support

* HCM team will manage

## Usage limitations, consumption count & invoicing

* Customers will be invoiced by employee
* Max number of documents per employee/user -> count/report (no hard limit in phase 1)
* Max number of users -> count/report active employees for invoicing purposes (no hard limit in phase 1)
  + Phase1: Standard list of users (Content Admin dashboard with Solution Admin role) Report from Keycloak report to be retrieved manually by NTT Data operator (e.g. once a month) showing current number of users per customer & tenant
  + Phase 2: integration with Snowflake/Titan reporting solution for it.products currently covering it.capture cloud
    - DataFactory in Azure will be set up to pull relevant information and pass it to Snowflake, where reports are defined that support invoice generation by the admin teams
      * Number of employees
      * Number of Documents per tenant/customer
      * Average number of documents per employee
      * Average retention time of documents

## Backup strategy

* Rolling backup schedule for it.archive databases (tbd)
* Backup & Restore of documents and Yuuvis configuration is handled by Optimal Systems according to contract (tbd)

## Availability/uptime and fallbacks

* it.archive tbd
* Yuuvis (see contract)

## Performance and scaling

To ensure sufficient performance, the solution will be monitored and resources will be tuned manually

* Key response times for it.archive will be monitored via Azure AppInsights
  + it.archive client AppService
  + it.archive client to backend
  + Repeated Yuuvis connection errors
  + Repeated SF connection errors
  + (phase 2) Yuuvis & SF connection speed (response times beyond threshold)
* (phase 2) Alerts to be configured tbd
* Key response times / up-time of Yuuvis and sufficient scaling will be managed by Optimal Systems according to contract