

PUBLIC

SAP Medical Research Insights 3.0

Document Version: 1.0.1 – 2019-04-01

Extensibility Guide

© 2019 SAP SE or an SAP affiliate company. All rights reserved.

THE BEST RUN SAP

Content

1 SAP Medical Research Insights Extensibility Guide. 3

2 Getting Started. 4

2.1 Use Cases. 4

2.2 Technical Prerequisites. 5

2.3 Required Knowledge. 6

2.4 Technical Overview. 7

2.5 Extension Process. 8

3 Integrating Custom Visualizations into Patient Analytics. 10

3.1 Creating Your Custom JSON Schema. 10

3.2 Uploading Your JSON Schema. 12

3.3 Mapping JSON Entities to Attributes. 15

3.4 Creating Custom Visualization Extensions Using SAPUI5. 17

3.5 Creating Custom Visualization Extensions Without SAPUI5. 18

3.6 Building and Deploying Extensions. 20

3.7 Integrating Custom Visualization Extensions. 22

3.8 Accessing Custom Visualizations. 23

4 Analyzing Patient Analytics Data in External Applications. 24

4.1 Exposing Data Using the Data Access API. 26

Creating Your Custom JSON Schema. 26

Uploading Your JSON Schema. 28

Mapping JSON Entities to Attributes. 31

Defining and Saving Filters for the Data Access API. 33

Generating the Data Access API. 34

Loading and Using the Data Exposed by the API. 34

4.2 Tutorial: Exposing Data from Patient Analytics for Analysis in R. 36

Preparing Data to Expose to R. 37

Exposing Data from Patient Analytics for Analysis in R. 40

Loading Data Exposed for Analysis into R. 43

5 Security Considerations. 47

1 SAP Medical Research Insights Extensibility Guide

This guide provides information about how to extend SAP Medical Research Insights and is intended primarily for developers and consultants who want to extend the solution. It is also a reference for researchers and clinicians working with extensions or features to support extensions in *Patient Analytics*.

> **i** Note
>
> Any extensions to the solution that are developed using the methods outlined in this guide are not related to extensions or plug-ins within the plug-in framework of the SAP Connected Health platform. The SAP Connected Health plug-in framework is not used to implement these extensions and the extensions developed are not listed or activated in the *Plug-In Cockpit* of SAP Connected Health.

2 Getting Started

Learn about when you might want to extend SAP Medical Research Insights and what you need to know and do before you extend the solution.

Related Information

Use Cases [page 4]
Technical Prerequisites [page 5]
Required Knowledge [page 6]
Technical Overview [page 7]
Extension Process [page 8]

2.1 Use Cases

Understand when you might want to extend SAP Medical Research Insights.

With SAP Medical Research Insights, you can analyze patient data using the available charts and visualizations. But you may also want to extend the solution to support the following scenarios:

- Integrate your own custom algorithms and visualizations so users can work with visualizations not supported in the existing charts.
  An example of this might be developing a pie chart that you integrate into *Patient Analytics* and is then available for use in the same way as the standard charts.
- Analyze data from *Patient Analytics* in an external or a third-party application and visualize the analysis results in that application.
  An example of this might be developing an odds ratio chart or a Forest plot chart in a third-party tool, such as R, and exposing and consuming patient data from *Patient Analytics* for analysis in this external chart.

You can support these scenarios using the new extensibility features of SAP Medical Research Insights. These features allow you to extend your use of *Patient Analytics*, the main end-user application of SAP Medical Research Insights.

Related Information

Extension Process [page 8]

2.2 Technical Prerequisites

To extend the analysis features of SAP Medical Research Insights, there are technical prerequisites that you must meet.

Before extending the features of SAP Medical Research Insights, ensure that the following requirements are met:

Required to Extend SAP Medical Research Insights

- You've installed SAP Medical Research Insights 3.0 on SAP Connected Health 2.0.
- You've installed SAP HANA 2.0 SPS03, enterprise edition.

> **i Note**
>
> A runtime edition of SAP HANA is not sufficient when you want to extend the features of SAP Medical Research Insights.

- You've installed Java Runtime Environment (JRE) 8 on your local development machine.
- You've installed Node.js v8.9.4 or higher on your local development machine.

Required to Use R as Data Analysis Extension to SAP Medical Research Insights

- You've installed RStudio on your local machine (on which you want to analyze SAP Medical Research Insights data externally).
- You're using R version 3.4.4 or higher.
- You've installed the following R packages:
  - `httr`
  - `jsonlite`
  - `dplyr`
  - `tidyverse`
  - `sjPlot` (if you want to visualize effects plots of regression models, for example)

> **i Note**
>
> These packages are publicly available in the R package repository hosted by the Comprehensive R Archive Network (CRAN).

Steps to Complete Before You Start Developing

The SAP public NPM (Node Package Manager) registry is the SAP registry for official SAP packages. We recommend that you use the `npm` command to configure this SAP registry as your target registry. Do this by executing the following command on your local development machine on which you are planning to develop your extensions:

```
npm config set @sap:registry https://npm.sap.com
```

2.3 Required Knowledge

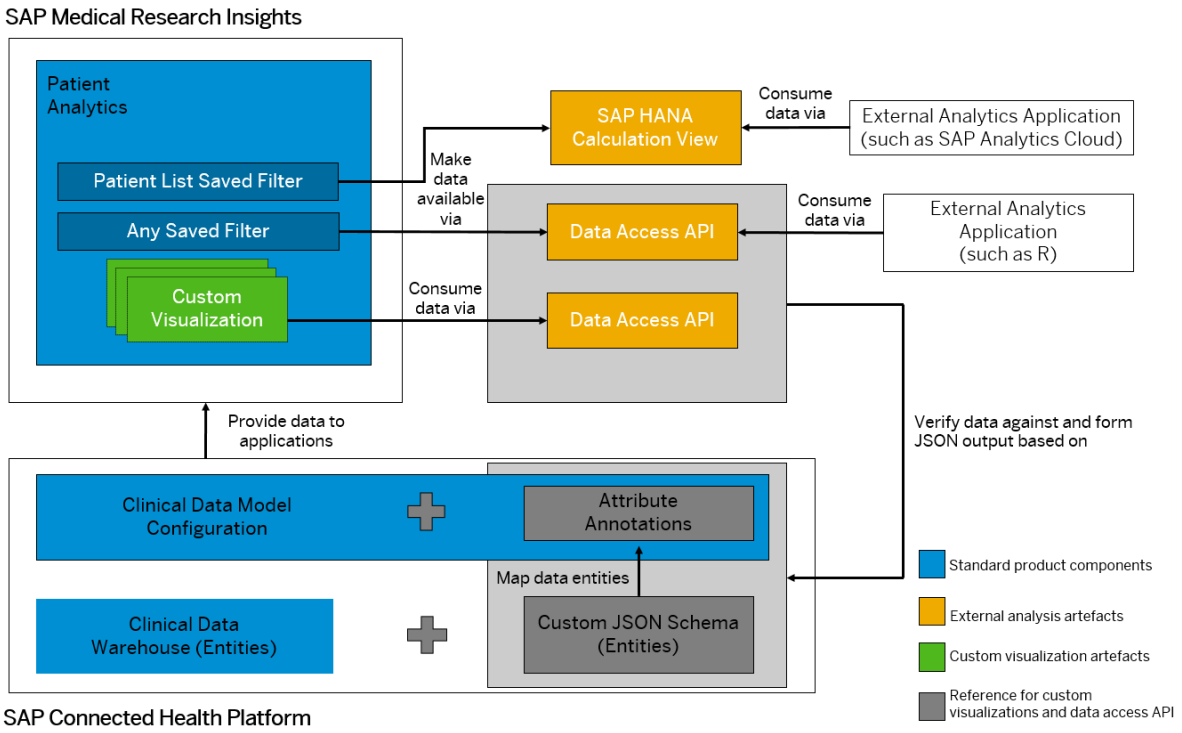
Before continuing with this guide, ensure that you have at least a basic understanding of certain development frameworks, programming languages, and SAP offerings.

Required Knowledge	Activities Requiring Knowledge	Reference
SAP Medical Research Insights	All activities relating to the extension framework assume knowledge of the features and system landscape of SAP Medical Research Insights	SAP Medical Research Insights
SAP HANA platform 2.0 SPS3 (on-premise), and specifically: • An understanding of calculation views • An understanding of SQL queries	• Working with and consuming data from calculation views when data is exposed from *Patient Analytics* in a calculation view • Running SQL queries or writing your own queries when developing a custom extension with or without SAPUI5	• SAP HANA Platform • Calculation Views (SAP HANA Developer Guide) • Introduction to SQL (SAP HANA SQL and System Views Reference)
SAP HANA XS Advanced (including knowledge of using the MTA Archive (MTAR) builder)	Developing, building, and deploying a custom extension developed with or without SAPUI5	• SAP HANA Developer Guide for XS Advanced Model • The MTA Archive (MTAR) Builder
SAP HANA development tools	Developing, building, and deploying a custom extension developed with or without SAPUI5	• SAP HANA Studio or • SAP HANA Web-Based Development Workbench
SAPUI5 / OpenUI5	Developing a custom extension with SAPUI5	OpenUI5 Developer Documentation
FHIR standard and FHIR definitions in JSON format (Release 3 (STU3))	Creating and uploading the custom JSON schema that is required to implement custom visualizations within *Patient Analytics* and also to allow data to be exposed using the data access API	https://www.hl7.org/fhir/

Required Knowledge	Activities Requiring Knowledge	Reference
Knowledge of working with an HTTP client to generate OAuth 2.0 tokens and send HTTP requests	Creating and uploading the custom JSON schema that is required to implement custom visualizations within *Patient Analytics* and also to allow data to be exposed using the data access API	Any HTTP client of your choice
JavaScript and Node.js v8.9.4	Developing a custom extension with or without SAPUI5	Any JavaScript and Node.js tutorials or learning materials
Knowledge of the external or third-party solution or analytic tool that you want to use to consume data from *Patient Analytics*, and an understanding of how it consumes and displays data	Consuming and analyzing exposed data in external tool	Applicable reference material specific to the solution or tool that you want to use as a means of extending data analysis

2.4 Technical Overview

The following figure provides a technical overview and outlines the data flow of the SAP Medical Research Insights extension framework.



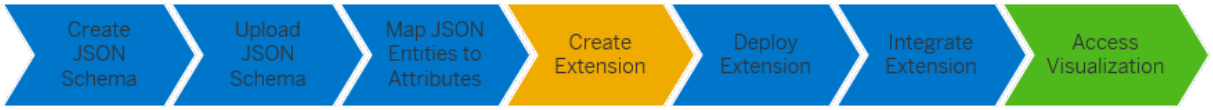
2.5 Extension Process

Understand the high-level steps involved in using the extensibility features of SAP Medical Research Insights.

The process to extend SAP Medical Research Insights varies depending on your extension scenario. Different steps are involved depending on whether you want to integrate your own custom visualization within *Patient Analytics* or analyze data from *Patient Analytics* in an external or third-party application. The processes for each of these scenarios are outlined here. The roles of the persons responsible for each step are color-coded as follows:

- **Blue**: Administrator
- **Orange**: Developer
- **Green**: Clinician/data analyst

Scenario 1: Integrating Custom Visualizations into Patient Analytics



1. Create your custom JSON schema to reflect your data model.
2. Upload your custom JSON schema to the server on which SAP Medical Research Insights is installed.
3. In the *Clinical Data Model Configuration* application of SAP Connected Health, add annotations to the attributes that you want to expose. These annotations allow you to map these attributes to your JSON schema entities.
4. Develop your SAPUI5 or non-SAPUI5 custom visualization extension as required.
5. Deploy your custom visualization extension.
6. Set up a configuration in the *Patient Analytics Configuration* application to integrate your visualization as an additional chart type for your users.
7. Access your custom visualization in *Patient Analytics*.

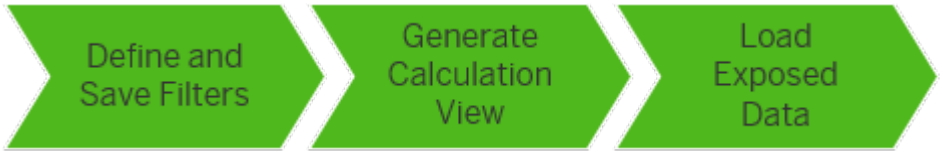
Scenario 2a: Analyzing Patient Analytics Data in External Applications - Using the Data Access API



1. Create your custom JSON schema to reflect your data model.

2. Upload your custom JSON schema to the server on which SAP Medical Research Insights is installed.
3. In the *Clinical Data Model Configuration* application of SAP Connected Health, add annotations to the attributes that you want to expose. These annotations allow you to map these attributes to your JSON schema entities.
4. Define your filters in *Patient Analytics* by restricting your patient set to those patients of interest. Then save your filter settings.
5. From the filter you saved in *Patient Analytics*, generate a URL for the data access API.
6. Load the exposed data from *Patient Analytics* by using the data access API URL generated to call the API in your external or third-party application.

Scenario 2b: Analyzing Patient Analytics Data in External Applications - Using an SAP HANA Calculation View



> **i** Note
>
> The option to create a calculation view can be selected only for saved filters that you created based on the **patient list**. You add columns to your patient list for each of the interactions and attributes that you want to be included in the exposed data. You can't select this option for any saved filters that you created based on other chart types.

1. Define your filters in *Patient Analytics* by restricting your patient set to those patients of interest and saving your filter setting:
   - Apply your filters as required to filter your patient set.
   - Switch to the patient list.
   - Add columns for the interactions and attributes that you want to be included in the exposed data.
   - Save your filtering and patient list settings as a saved filter.
2. From the patient list filter you saved in *Patient Analytics*, generate an SAP HANA calculation view for your data.
3. Load the exposed data from *Patient Analytics* by consuming the SAP HANA calculation view in your external application.

Related Information

Use Cases [page 4]

3 Integrating Custom Visualizations into Patient Analytics

Using the extensibility framework, you can develop extensions to integrate your own algorithms and accompanying custom visualizations into Patient Analytics.

3.1 Creating Your Custom JSON Schema

The data access API of SAP Medical Research Insights is loosely based on FHIR Release 3 (STU3) and those interaction types and attributes in your data model that you configure as FHIR-like entities. You create a custom JSON schema that defines these entities and you subsequently map these entities to interaction types and attributes in your clinical data model configuration.

Context

You create a custom JSON schema with predefined entities that is used as a basis to define what data you can expose from SAP Medical Research Insights. Before you can work with the data access API or custom visualizations, you must have defined this custom schema to reflect your data model. You customize the schema based on your data model and the clinical data model configurations that you use to consume and expose data in SAP Medical Research Insights. This JSON schema is what enables SAP Medical Research Insights to retrieve the data that you want to expose from the database and format it as FHIR-like data.

Procedure

1. Download the FHIR JSON schema (STU3) from the HL7 FHIR Web site.

   → Tip

   You can download the JSON schema zip, for example, from https://hl7.org/fhir/stu3/downloads.html .

   Unzip the `definitions.json.zip` that you download. Then unzip `fhir.schema.json.zip`. The full JSON schema is in the file `fhir.schema.json`.

2. Use the FHIR JSON schema as a reference for creating your custom JSON schema. You can tailor your schema to your data model by copying the content and modifying it in your custom file, for example. Resources that you might typically require include entities such as patient, observation, and encounter resources.

Example

Imagine you have a chemotherapy interaction defined in your clinical data model configuration. To retrieve the chemotherapy data to be exposed and later consumed in an external application, you first define this interaction as a custom entity. You do so by extending the `procedure` entity that already exists in the standard JSON schema. You would add the attributes of the chemotherapy interaction, such as `opsCode` and `protocol` as well as other attributes, to the JSON schema as follows:

1. In your custom JSON schema, add a chemotherapy entity within the `definitions` element.
2. Then add properties to this entity to reflect the attributes that you want to expose for this interaction, for example:

```
                "Chemotherapy": {
                    "description": "Measurements and simple assertions
made about a patient, device or other subject.",
                    "properties": {
                        "resourceType": {
                            "description": "This is a Chemotherapy
resource",
                            "const": "Chemotherapy"
                        },
                        "id": {
                            "description": "The logical id of the
resource, as used in the URL for the resource. Once assigned, this value
never changes.",
                            "$ref": "#/definitions/id"
                        },
                        "identifier": {
                            "description": "A unique identifier assigned
to this observation.",
                            "items": {
                                "$ref": "#/definitions/Identifier"
                            },
                            "type": "array"
                        },
                        "startDateTime": {
                            "description": "The time or time-period the
observed value is asserted as being true. For biological subjects - e.g.
human patients - this is usually called the \"physiologically relevant time
\". This is usually either the time of the procedure or of specimen
collection, but very often the source of the date/time is not known, only
the date/time itself.",
                            "pattern": "^([0-9]([0-9]([0-9][1-9]|[1-9]0)|
[1-9]00)|[1-9]000)(-(0[1-9]|1[0-2])(-(0[1-9]|[1-2][0-9]|3[0-1])(T([01]
[0-9]|2[0-3]):[0-5][0-9]:([0-5][0-9]|60)(\\.[0-9]+)?(Z|(\\+|-)((0[0-9]|
1[0-3]):[0-5][0-9]|14:00)))?)?)?$",
                            "type": "array"
                        },
                        "endDateTime": {
                            "description": "The time or time-period the
observed value is asserted as being true. For biological subjects - e.g.
human patients - this is usually called the \"physiologically relevant time
\". This is usually either the time of the procedure or of specimen
collection, but very often the source of the date/time is not known, only
the date/time itself.",
                            "pattern": "^([0-9]([0-9]([0-9][1-9]|[1-9]0)|
[1-9]00)|[1-9]000)(-(0[1-9]|1[0-2])(-(0[1-9]|[1-2][0-9]|3[0-1])(T([01]
[0-9]|2[0-3]):[0-5][0-9]:([0-5][0-9]|60)(\\.[0-9]+)?(Z|(\\+|-)((0[0-9]|
1[0-3]):[0-5][0-9]|14:00)))?)?)?$",
                            "type": "array"
                        },
                        "opsCode": {
                            "description": "OPS Code",
                            "$ref": "#/definitions/Reference",
```

```
                        "type": "array"
                    },
                    "protocol": {
                        "description": "Protocol",
                        "$ref": "#/definitions/Reference",
                        "type": "array"
                    }
                },
                "additionalProperties": false,
                "required": [
                    "code",
                    "resourceType"
                ]
            }
```

> → Tip
>
> To ensure the data access API works correctly, do not use spaces in the names of your entities or properties. Also be aware that names are handled as case sensitive.

3. Validate your JSON schema using a JSON validator of your choice.
4. Once you've customized and validated your JSON schema, you're ready to continue with the next steps:
    1. Upload your schema to the server on which SAP Medical Research Insights is installed. For more information, see Uploading Your JSON Schema [page 12].
    2. Map the `opsCode` and `protocol` attributes of your *Chemotherapy* interaction in your clinical data model configuration to the schema entity by adding an annotation to each attribute. For more information, see Mapping JSON Entities to Attributes [page 15].

3.2 Uploading Your JSON Schema

Before you work with the data access API or custom visualizations, you upload your custom JSON schema to the server on which SAP Medical Research Insights is installed.

Context

You've created a custom JSON schema for your data model. You're ready to upload this schema to the SAP HANA server, on which SAP Medical Research Insights is installed, using a POST request. You're uploading this file so you can then map the attributes of your data model to your FHIR-like JSON entities. You use your custom schema as a basis for this mapping. And the system uses this schema as a reference to form the JSON output as the response to API calls.

> ℹ Note
>
> You can upload one JSON schema per system instance. If you subsequently upload another JSON schema, this schema automatically overwrites the existing schema. The JSON schema that you upload applies to all users of the system instance. So you don't have to assign it to specific users or specific configurations.

Procedure

1. To get the information you require to upload your JSON schema to the server, start by identifying the location of your SAP Medical Research Insights installation. Do so by logging on to the SAP HANA XS advanced instance where SAP Medical Research Insights is installed, and running commands to obtain the `urls`.
   a. From your command line, use the following command to log on to your SAP HANA XS advanced instance as administrator, such as XSA_ADMIN user:

   ```
   xs login -a https://<hostname>:3<instance number>30 -u <username> -p
   <password> --cacert </path/to/default.root.crt.pem>
   ```

   > **i Note**
   >
   > The client needs a trust certificate to establish a secure connection with the server. The path here denotes the location of the file `default.root.crt.pem` on the client side.
   >
   > You can download this certificate from the SAP HANA server. The certificate `default.root.crt.pem` is typically installed in the following default location:
   >
   > `<installation_path>/<SID>/xs/controller_data/controller/ssl-pub/router/default.root.crt.pem`

   b. Switch to the organization and space in which SAP Medical Research Insights is installed and list all applications installed in that space by running the following commands:

   ```
   xs t -o <org name> -s <space name>
   xs a
   ```

   c. Display the information for the application `mri-pa-svc` by entering the following command:

   ```
   xs app mri-pa-svc
   ```

   d. In the details displayed, make a note of the value shown for `urls`.
2. Display all the environment variables set for the `mri-pa-svc` application so that you can identify what's set for the client ID and client secret.
   a. Run the following command to display all environment variables set:

   ```
   xs env mri-pa-svc
   ```

   b. From the JSON output that displays the variables set, navigate to the `"xsuaa"` element and make a note of the values shown for the `"clientid"` and `"clientsecret"` variables.
3. Using an HTTP client of your choice, request an `OAuth 2.0` authorization token that enables you to send an HTTP POST request. Do this using the following information:

Information in Request	Description or Information Source
Token name	Can be freely defined
Grant type	Password credentials
Access token URL	`https://<hostname>:3<instance number>32/uaa-security/oauth/token`

Information in Request	Description or Information Source
Username	Name of the user you use to access SAP Medical Research Insights
Password	Password for your SAP Medical Research Insights user
Client ID	The value noted for `"clientid"` in step 2
Client secret	The value noted for `"clientsecret"` in step 2

4. Still in your HTTP client, use the access token that you've generated to send a POST request, with your JSON schema as the body, to your instance.
   a. Locate your custom JSON schema that you tailored to match your data model.
   b. Open the JSON file in a suitable editor and insert a new first line with `{"schema":`. Also insert the corresponding closing `}` as the final line.
   c. Paste the contents of this JSON file into the area of your HTTP client that contains the body for the POST request.

      > → Tip
      >
      > For an example of the custom part of a JSON file, see Preparing Data to Expose to R [page 37].

   d. Enter the following information for the request header:

Information in Request	Description or Information Source
URL for POST	`https://<hostname>:<port>/sap/hc/mri/api/services/data/schema` You use the host name and port that you noted for `urls` for the `mri-pa-svc` application.
Content-Type	application/json

   e. When you've entered the body and header information, send the request.

      > i Note
      >
      > If, at any time, you want to check what JSON is currently uploaded to the server, you can also use the same end point above to retrieve the information with a GET request.

You've now uploaded your JSON schema to the SAP HANA server on which SAP Medical Research Insights is installed. Now that your JSON schema is in place, you can map the attributes of your data model to your FHIR-like JSON entities. You use your custom schema as a basis for this mapping. And the system uses this schema as a reference to form the JSON output as the response to API calls.

3.3 Mapping JSON Entities to Attributes

You map your JSON entities and their properties to the interaction attributes of your clinical data model. You do so by adding annotations to these attributes in your clinical data model configuration.

Context

You've created a custom JSON schema with FHIR-like entities and now want to map these entities to attributes in your clinical data model. To map your JSON entities to your clinical data model interaction attributes, you add an annotation to each of these attributes in the *Clinical Data Model Configuration* application.

> **i Note**
>
> The task outlined here assumes that you've already set up your clinical data model configuration. For more information about creating clinical data model configurations, see the *Clinical Data Model Configuration* section in the SAP Connected Health Platform Administration Guide.

Procedure

1. Go to the SAP Fiori launchpad for SAP Connected Health.

   The SAP Fiori launchpad is available at the following URL: `https://<WebServerHost>:3<SAPHANAinstance>30/go/chp-<space>`
2. Open the *Clinical Data Model Configuration* application.
3. Open a version of an existing configuration and edit the version.
4. For your basic data or each interaction that you want to allow to be used in a custom chart or analyzed externally, open the settings for the basic data or interaction under *Defined Interactions*.
5. For each attribute that you want to allow to be used in a custom chart or analyzed externally, open the settings for that attribute under *Basic Data* or *Saved Attributes*, and navigate to the *Annotations* field.
6. In the *Annotations* field, add an annotation in the following format:

   `data://<JSON entity>/<JSON property>`

   The entity and property names that you enter in your annotation must correspond with the entities and properties in your custom JSON schema.

   > **i Note**
   >
   > Entity and property names are case-sensitive. Each annotation that you enter must also be unique and configured just once.
7. Repeat steps 4–6 for all interactions and attributes that are to be available as data to be exposed or analyzed externally.

> **i Note**
>
> Always add an annotation to the patient ID attribute. This is an important key that must be exposed.

> **Example**
>
> The following table provides some examples of mappings between JSON entities and properties and clinical data model configuration interactions and attributes:

JSON Entity	JSON Property	Interaction	Attribute	Attribute Annotation
Patient	identifier	Basic Data	Patient ID	`data://Patient/identifier`
Patient	gender	Basic Data	Gender	`data://Patient/gender`
Vitalstatus	vitalStatus	Vital Status	Vital Status	`data://Vitalstatus/vitalStatus`
Chemotherapy	opsCode	Chemotherapy	OPS Code	`data://Chemotherapy/opsCode`
Chemotherapy	protocol	Chemotherapy	Protocol	`data://Chemotherapy/protocol`

8. Validate, activate, and save your configuration. Make note of the configuration name and version as you need this information in the next step.
9. Use this clinical configuration version in any *Patient Analytics* configurations to be used by clinicians or researchers who want to use a custom chart or analyze data externally.
   a. Use the new clinical data model configuration version in a new or existing *Patient Analytics* configuration.

      > **→ Remember**
      >
      > If you want users to be able to expose data for external analysis from saved filters, ensure that you activate the *Make Data Accessible Externally* option for the configuration or configurations assigned to the users.

      For more information about assigning configurations, see the *Configuring Patient Analytics* section of the SAP Medical Research Insights Administration Guide.
   b. Assign this *Patient Analytics* configuration to the relevant organization or users in the *Configuration Assignment* application.

      For more information about assigning configurations, see the *Assigning Configurations to Users* section of the SAP Connected Health Platform Administration Guide.

3.4 Creating Custom Visualization Extensions Using SAPUI5

Understand the process for how to create a custom visualization for *Patient Analytics* using SAPUI5.

Context

You are a developer and want to extend the features of SAP Medical Research Insights by developing a custom visualization using SAPUI5. You plan to integrate this visualization into *Patient Analytics*.

Procedure

1. Open your preferred IDE, such as SAP Web IDE for SAP HANA or SAP HANA Studio.
2. Create a SAPUI5 extension view, which involves creating a model view container and a controller file.
   a. Create a view XML file using the following code as a template. For the `controllerName` property, replace the `<extension_name>` placeholder with a descriptive name of your choice.

```
<mvc:View
    controllerName="sap.hc.mri.plugins.ui.<extension_name>"
    xmlns:mvc="sap.ui.core.mvc"
    xmlns:lib="sap.viz.ui5"
    xmlns:html="http://www.w3.org/1999/xhtml"
    xmlns="sap.m"
    xmlns:commons="sap.ui.commons"
    xmlns:appdata=
    "http://schemas.sap.com/sapui5/extension/sap.ui.core.CustomData/1"
    height="100%">
    <Page
        showHeader="false"
        showNavButton="false"
        enableScrolling="false"
        backgroundDesign="Solid"
        height="100%">
        <content><html:div id="chartArea" style="width:100%;height:
100%;"></html:div>
        </content>
    </Page>
</mvc:View>
```

   b. Similarly, create a controller file using the following code as a template. Replace the following placeholders:
      - `<extension_name>`: Replace with the same name you used for the view
      - `<entity>`: Replace with the entity that you want to retrieve data for, based on your custom JSON schema, such as the `Chemotherapy` entity
      - Also replace the final commented (`//`) lines with the relevant code

```
sap.ui.define([
  'jquery.sap.global',
  'sap/ui/core/mvc/Controller',
```

```
  'sap/ui/model/json/JSONModel',
], function (jQuery, Controller, JSONModel) {
    const chart =
Controller.extend('sap.hc.mri.plugins.ui.<extension_name>', {
    onInit: function () {
    this.eventBus = sap.ui.getCore().getEventBus();
    this.eventBus.subscribe('MRI_CUSTOM_CHART_REQUEST', this.updateChart,
this);

   },

    onExit: function () {
        this.eventBus.unsubscribe('MRI_CUSTOM_CHART_REQUEST',
this.updateChart, this);},
    updateChart: function (sChannelId, sEventId, oEventData) {
      if (oEventData && oEventData.data) {
          const mResponse = oEventData;
          const urlPathParam =
encodeURIComponent(JSON.stringify(this._mResponse.mriquery));
          const url = "/sap/hc/mri/api/services/data/"+urlPathParam+"/
<entity>"
          //For example: "/sap/hc/mri/api/services/data/
asdevberfygewuryfjhsdhadfadasd/Chemotherapy"

          // Send an AJAX GET request
          // Get the response and handle error scenarios
          // Modify the response data and update the data model that the
chart is bound to
        }
      }
    });
    return chart;
  });
```

> **i Note**
>
> You retrieve data for different entities (such as the patient or chemotherapy entities) individually. You can't retrieve data for different entities at the same time. You may therefore want to add multiple `const url` lines in your code to allow all relevant data to be retrieved.

You've now developed your SAPUI5 extension and are ready to build and deploy it.

3.5 Creating Custom Visualization Extensions Without SAPUI5

You might want to use a library other than what is available for SAPUI5 to implement your custom visualization. Understand the process for how to create a custom visualization for *Patient Analytics* without SAPUI5.

Context

You are a developer and want to extend the features of SAP Medical Research Insights by developing a custom visualization without using SAPUI5. You plan to integrate this visualization into *Patient Analytics*.

Procedure

1. Open your preferred IDE, such as SAP Web IDE for SAP HANA or SAP HANA Studio.
2. Create an HTML file using the template provided. This file is to be loaded later into *Patient Analytics* in an iframe.
   a. Import any required JavaScript libraries into the HTML file.
   b. Implement the listener **MRIPlugin.onMRIRequest**. The argument of this listener **e** has a `detail` property that contains an encoded request that can be used to send a request to the SAP HANA server. When you implement the listener, any change to filter cards causes the event listener to be triggered and your chart to be updated.
   c. Replace the `<entity>` with the entity that you want to retrieve data for, based on your custom JSON schema, such as the `Chemotherapy` entity.

```
<html>
    <head>
        <title>Patient Analytics Extension Example</title>
        <script type="text/javascript">
        var mriRequest = null;
        // Implement this to update your chart on every event change of
filter cards
        document.addEventListener("MRIPlugin.onMRIRequest", function(e){
           mriRequest = e.detail;
           const urlPathParam =
encodeURIComponent(JSON.stringify(mriRequest));
           const url = `/sap/hc/mri/api/services/data/${urlPathParam}/
<entity>`;
           //For example URL: /sap/hc/mri/api/services/data/
asdevberfygewuryfjhsdhadfadasd/Chemotherapy

          // Send an AJAX GET request
          // Handle the response
          // Example: document.getElementById("result").innerHTML =
JSON.stringify(response, null, 2);
          // Handle the error scenarios
        });
        // Use this to debug/examine the request URL path that is sent to
the SAP HANA server
        document.addEventListener("MRIPlugin.onMRIRequestDecoded",
function(e){
            document.getElementById("pRequest").innerHTML =
JSON.stringify(e.detail, null, 2);
        });
        </script>
    </head>
    <body>
      <div> Response from Server: </div>
      <pre id="result"></pre>
      <div> Patient Analytics Request: </div>
      <pre id="pRequest"></pre>
    </body>
</html>
```

   d. Optional: If you want to debug and examine the request sent to the `MRIPlugin.onMRIRequest`, you can implement the `MRIPlugin.onMRIRequestDecoded` listener. This listener gives you the full request in decoded form. It is essentially a representation of the filter cards and any constraints or controls that were applied in *Patient Analytics*.

You've now created an HTML file that is ready to be integrated into a *Patient Analytics* configuration.

3.6 Building and Deploying Extensions

Build and deploy your extension for SAP Medical Research Insights by generating an MTA archive and deploying this archive to your SAP HANA XS advanced instance.

Context

You've developed the SAPUI5 extension that you want to use in SAP Medical Research Insights and are ready to deploy it.

Procedure

1. Build an MTA archive (.mtar file) to deploy any extensions that you've created to your SAP HANA XS advanced instance.
   a. Copy the following code to create an MTA development descriptor (mta.yaml file).

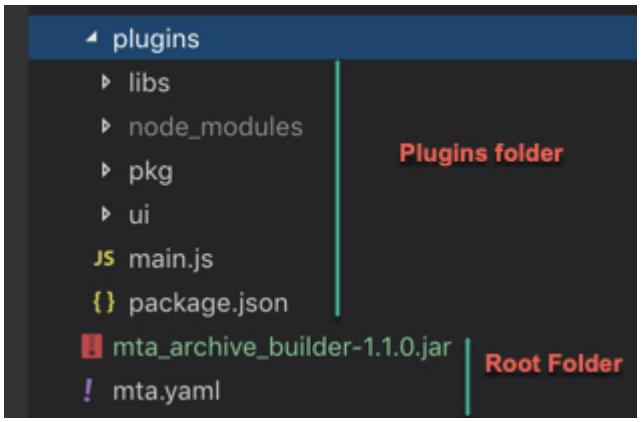
```
_schema-version: 2.0.0
ID: com.sap.mri.plugins
version: 3.1.0
modules:
  - name: mri-pa-ui-plugins
    type: nodejs
    path: plugins
    build-parameters:
      ignore: []
    requires:
      - name: chp-uaa
      - name: chp-hdi
    provides:
      - name: mripatpuiplugins
        properties:
           url: ${default-url}
           type: com.sap.chp.destination
           name: mripatpuiplugins
        public: true
      - name: mpaplugins
        properties:
          type: com.sap.chp.routes
          source: ^/sap/hc/mri/plugins(.*)$
          destination: mripatpuiplugins
          name: mpaplugins
        public: true
resources:
  - name: chp-uaa
    type: org.cloudfoundry.existing-service

  - name: chp-hdi
    type: org.cloudfoundry.existing-service
```

   b. If you don't already have it, download the multi-target application archive (MTAR) builder:
      - Navigate to SAP Development Tools under https://tools.hana.ondemand.com/#cloud.
      - Select the latest version of the .jar or .zip file containing the multi-target application archive builder, and download it to a local folder.

> → Tip
>
> Download the **.zip** file to avoid browser warnings about potentially dangerous content. Unzip the .zip file to get the **.jar** file.

c. Copy the .jar file for the archive builder to the root folder of your project. Your folder structure should then look as follows:



d. Build your archive (.mtar file).

- When all your files are in place as in the example shown, navigate to your project root folder using the terminal.
- Run the following command, replacing the placeholders as described:

```
java -jar mta_archive_builder-<version_number>.jar --build-target=XSA --mtar <file_name>.mtar build
```

  - `<version_number>`: Replace with the version number indicated in the .jar file name
  - `<file_name>`: Replace with a descriptive name of your choice
- When the command has finished executing, verify that the .mtar file had been generated in your root folder.

2. Deploy the MTA archive (.mtar) that you've just built to your SAP HANA XS advanced instance.

a. From your command line, use the following command to log on to your SAP HANA XS advanced instance as administrator, such as XSA_ADMIN user:

```
xs login -a https://<hostname>:3<instance number>30 -u <username> -p <password> --cacert </path/to/default.root.crt.pem>
```

> i Note
>
> The client needs a trust certificate to establish a secure connection with the server. The path here denotes the location of the file `default.root.crt.pem` on the client side.
>
> You can download this certificate from the SAP HANA server. The certificate `default.root.crt.pem` is typically installed in the following default location:
>
> `<installation_path>/<SID>/xs/controller_data/controller/ssl-pub/router/default.root.crt.pem`

b. Deploy your MTA archive by executing the following command:

```
xs deploy <file_name>.mtar
```

You've now built and deployed your MTA archive, which is ready to be integrated into a *Patient Analytics* configuration.

Related Information

The MTA Archive (MTAR) Builder

3.7 Integrating Custom Visualization Extensions

A custom visualization extension developed with SAPUI5 or another non-SAPUI5 technology is made available to users by integrating it into a *Patient Analytics* configuration.

Context

You or a developer has built and deployed an MTA archive containing a custom visualization. This extension is now ready to be integrated into a *Patient Analytics* configuration and enabled for use.

Procedure

1. Go to the SAP Fiori launchpad for SAP Medical Research Insights.

   The SAP Fiori launchpad is available at the following URL: `https://<WebServerHost>:3<SAPHANAinstance>30/go/chp-<space>`
2. Launch the *Patient Analytics Configuration* application.
3. Select the data model configuration on which you want to base your *Patient Analytics* configuration.
4. Add a new configuration or choose an existing *Patient Analytics* configuration.
5. On the *Charts* tab, navigate to *Chart Types* > *Custom Visualization*.
6. Choose *Add Visualization*.
7. Specify the following details:

Visualization Name	Specifies the name of the visualization, which appears on the user interface
Visualization Type	Specifies whether the custom visualization uses a SAPUI5 component or iframe

Visualization View Source	Specifies the location of the visualization view, expressed as either: **`sap.hc.mri.plugins.ui.<chart-name>`** (for SAPUI5 components) or **`sap/hc/mri/plugins/ui/<file-name>.html`** (for iframes)

8. Save your *Patient Analytics* configuration.

   You've now integrated your custom visualization into a *Patient Analytics* configuration that can be assigned to specific users or to an organization.

3.8 Accessing Custom Visualizations

As a clinician or researcher, you access custom visualizations that have been developed for you from the chart area in *Patient Analytics*.

Context

A developer in your organization has developed a custom visualization or chart to visualize results that cannot be visualized using the standard chart types in *Patient Analytics*. Before you can access a custom visualization, your system administrator must have done the following:

- Integrated the custom visualization within a *Patient Analytics* configuration
- Assigned the *Patient Analytics* configuration that integrates the visualization to you or your organization

Procedure

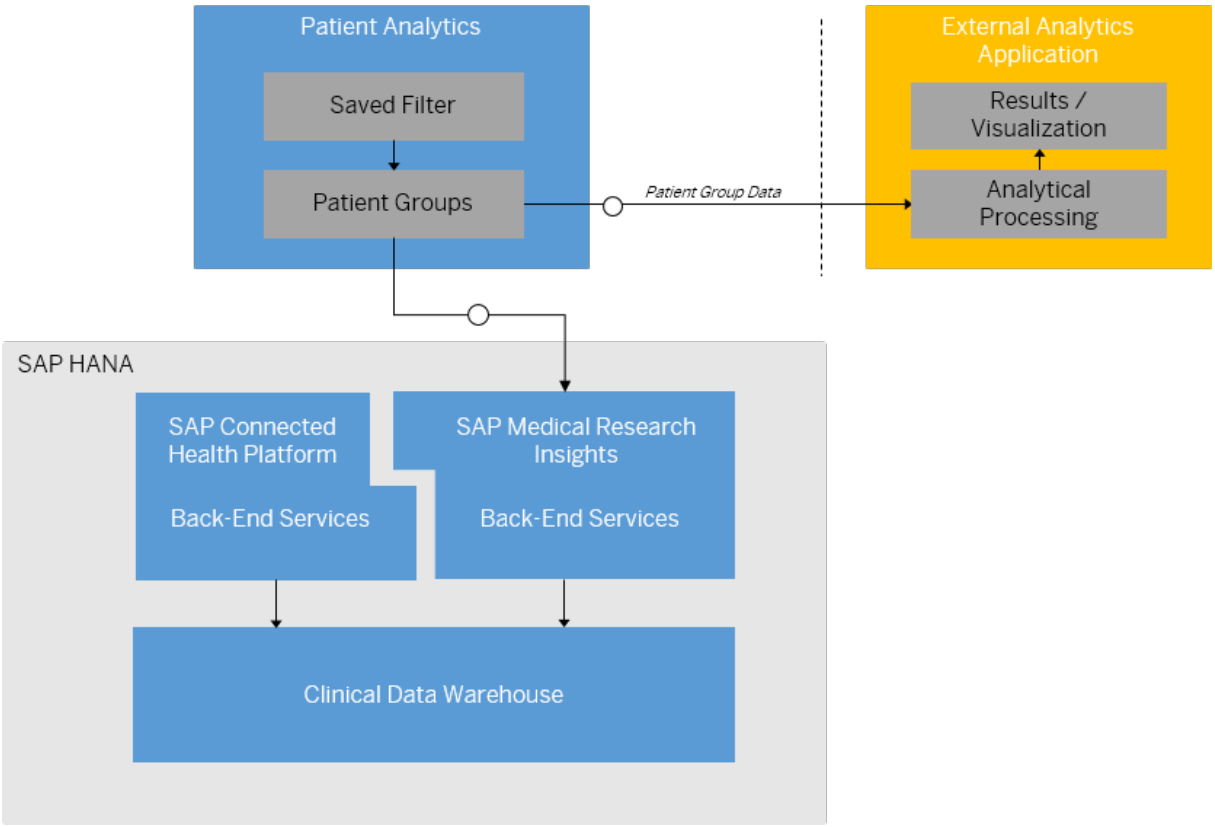
1. Go to the SAP Fiori launchpad for SAP Medical Research Insights.

   The SAP Fiori launchpad is available at the following URL: `https://<WebServerHost>:3<SAPHANAinstance>30/go/chp-<space>`
2. Open the *Patient Analytics* application.
3. In the chart area of *Patient Analytics*, choose the chart type *(Custom Visualization)*.
   Within the chart area, a separate tab is displayed for each custom visualization that has been developed and integrated for you. The name of each extension is indicated in each of the tab titles.
4. Choose the tab for the custom visualization that you want to work with. For information about how to work with the custom visualizations developed for you, contact your system administrator or developer.

4 Analyzing Patient Analytics Data in External Applications

Use the options available in *Patient Analytics* to expose data for analysis in external applications.

The following figure outlines the system landscape components involved:



As a user of *Patient Analytics*, you can expose patient data for analysis in external or third-party solutions by using the *Make Data Accessible Externally* option for a saved filter in Patient Analytics. So the filters that you've set and then saved as a saved filter determine which data that you can expose. The filters that form a saved filter define the patient group for which data is to be exposed. In this section, this set of filters that form a saved filter is referred to as a 'patient group definition'.

> **i Note**
>
> From external applications, you can only access patient data that you're authorized to access.

The *Make Data Accessible Externally* feature allows you to expose data via a saved filter by means of two access points:

Access Point	Description	Use Case
Data Access API	• You choose *Data Access API* to generate a URL for a FHIR-like API. **i Note** Before you use this option, you must have added annotations to your data attributes within your clinical data model configuration. These annotations correspond to the entities in your custom JSON schema.	You can use the URL generated with this option to call the API directly from third-party tools such as R.
Calculation View	• You choose *Calculation View* > *Generate Calculation View* to generate an SAP HANA calculation view. **i Note** The option to create a calculation view can be selected only for saved filters that you created based on the **patient list**. You add columns to your patient list for each of the interactions and attributes that you want to be included in the exposed data. You can't select this option for any saved filters that you created based on other chart types.	External applications such as SAP Analytics Cloud can consume this SAP HANA calculation view generated from *Patient Analytics*.

i Note

When patient data is exposed for analysis, the security settings of SAP Medical Research Insights and SAP Connected Health platform apply. This means that patient data is exposed only if the person exposing it is authorized to view that data, and this data is assigned to the same organizational unit as the person exposing it.

Related Information

Exposing Data Using the Data Access API [page 26]
Tutorial: Exposing Data from Patient Analytics for Analysis in R [page 36]

4.1 Exposing Data Using the Data Access API

The data access API of SAP Medical Research Insights is one of the options available for exposing data from *Patient Analytics* for analysis in external applications, such as R.

4.1.1 Creating Your Custom JSON Schema

The data access API of SAP Medical Research Insights is loosely based on FHIR Release 3 (STU3) and those interaction types and attributes in your data model that you configure as FHIR-like entities. You create a custom JSON schema that defines these entities and you subsequently map these entities to interaction types and attributes in your clinical data model configuration.

Context

You create a custom JSON schema with predefined entities that is used as a basis to define what data you can expose from SAP Medical Research Insights. Before you can work with the data access API or custom visualizations, you must have defined this custom schema to reflect your data model. You customize the schema based on your data model and the clinical data model configurations that you use to consume and expose data in SAP Medical Research Insights. This JSON schema is what enables SAP Medical Research Insights to retrieve the data that you want to expose from the database and format it as FHIR-like data.

Procedure

1. Download the FHIR JSON schema (STU3) from the HL7 FHIR Web site.

   → Tip

   You can download the JSON schema zip, for example, from https://hl7.org/fhir/stu3/downloads.html.

   Unzip the `definitions.json.zip` that you download. Then unzip `fhir.schema.json.zip`. The full JSON schema is in the file `fhir.schema.json`.

2. Use the FHIR JSON schema as a reference for creating your custom JSON schema. You can tailor your schema to your data model by copying the content and modifying it in your custom file, for example. Resources that you might typically require include entities such as patient, observation, and encounter resources.

   Example

   Imagine you have a chemotherapy interaction defined in your clinical data model configuration. To retrieve the chemotherapy data to be exposed and later consumed in an external application, you first define this interaction as a custom entity. You do so by extending the `procedure` entity that already exists in the

standard JSON schema. You would add the attributes of the chemotherapy interaction, such as `opsCode` and `protocol` as well as other attributes, to the JSON schema as follows:

1. In your custom JSON schema, add a chemotherapy entity within the `definitions` element.
2. Then add properties to this entity to reflect the attributes that you want to expose for this interaction, for example:

```
                "Chemotherapy": {
                    "description": "Measurements and simple assertions
made about a patient, device or other subject.",
                    "properties": {
                        "resourceType": {
                            "description": "This is a Chemotherapy
resource",
                            "const": "Chemotherapy"
                        },
                        "id": {
                            "description": "The logical id of the
resource, as used in the URL for the resource. Once assigned, this value
never changes.",
                            "$ref": "#/definitions/id"
                        },
                        "identifier": {
                            "description": "A unique identifier assigned
to this observation.",
                            "items": {
                                "$ref": "#/definitions/Identifier"
                            },
                            "type": "array"
                        },
                        "startDateTime": {
                            "description": "The time or time-period the
observed value is asserted as being true. For biological subjects - e.g.
human patients - this is usually called the \"physiologically relevant time
\". This is usually either the time of the procedure or of specimen
collection, but very often the source of the date/time is not known, only
the date/time itself.",
                            "pattern": "^([0-9]([0-9]([0-9][1-9]|[1-9]0)|
[1-9]00)|[1-9]000)(-(0[1-9]|1[0-2])(-(0[1-9]|[1-2][0-9]|3[0-1])(T([01]
[0-9]|2[0-3]):[0-5][0-9]:([0-5][0-9]|60)(\\.[0-9]+)?(Z|(\\+|-)((0[0-9]|
1[0-3]):[0-5][0-9]|14:00)))?)?)?$",
                            "type": "array"
                        },
                        "endDateTime": {
                            "description": "The time or time-period the
observed value is asserted as being true. For biological subjects - e.g.
human patients - this is usually called the \"physiologically relevant time
\". This is usually either the time of the procedure or of specimen
collection, but very often the source of the date/time is not known, only
the date/time itself.",
                            "pattern": "^([0-9]([0-9]([0-9][1-9]|[1-9]0)|
[1-9]00)|[1-9]000)(-(0[1-9]|1[0-2])(-(0[1-9]|[1-2][0-9]|3[0-1])(T([01]
[0-9]|2[0-3]):[0-5][0-9]:([0-5][0-9]|60)(\\.[0-9]+)?(Z|(\\+|-)((0[0-9]|
1[0-3]):[0-5][0-9]|14:00)))?)?)?$",
                            "type": "array"
                        },
                        "opsCode": {
                            "description": "OPS Code",
                            "$ref": "#/definitions/Reference",
                            "type": "array"
                        },
                        "protocol": {
                            "description": "Protocol",
                            "$ref": "#/definitions/Reference",
                            "type": "array"
                        }
```

```
                    },
                    "additionalProperties": false,
                    "required": [
                        "code",
                        "resourceType"
                    ]
                }
```

> → Tip
>
> To ensure the data access API works correctly, do not use spaces in the names of your entities or properties. Also be aware that names are handled as case sensitive.

3. Validate your JSON schema using a JSON validator of your choice.
4. Once you've customized and validated your JSON schema, you're ready to continue with the next steps:
    1. Upload your schema to the server on which SAP Medical Research Insights is installed. For more information, see Uploading Your JSON Schema [page 12].
    2. Map the `opsCode` and `protocol` attributes of your *Chemotherapy* interaction in your clinical data model configuration to the schema entity by adding an annotation to each attribute. For more information, see Mapping JSON Entities to Attributes [page 15].

4.1.2 Uploading Your JSON Schema

Before you work with the data access API or custom visualizations, you upload your custom JSON schema to the server on which SAP Medical Research Insights is installed.

Context

You've created a custom JSON schema for your data model. You're ready to upload this schema to the SAP HANA server, on which SAP Medical Research Insights is installed, using a POST request. You're uploading this file so you can then map the attributes of your data model to your FHIR-like JSON entities. You use your custom schema as a basis for this mapping. And the system uses this schema as a reference to form the JSON output as the response to API calls.

> ℹ Note
>
> You can upload one JSON schema per system instance. If you subsequently upload another JSON schema, this schema automatically overwrites the existing schema. The JSON schema that you upload applies to all users of the system instance. So you don't have to assign it to specific users or specific configurations.

Procedure

1. To get the information you require to upload your JSON schema to the server, start by identifying the location of your SAP Medical Research Insights installation. Do so by logging on to the SAP HANA XS advanced instance where SAP Medical Research Insights is installed, and running commands to obtain the `urls`.
   a. From your command line, use the following command to log on to your SAP HANA XS advanced instance as administrator, such as XSA_ADMIN user:

   ```
   xs login -a https://<hostname>:3<instance number>30 -u <username> -p
   <password> --cacert </path/to/default.root.crt.pem>
   ```

   > **i Note**
   >
   > The client needs a trust certificate to establish a secure connection with the server. The path here denotes the location of the file `default.root.crt.pem` on the client side.
   >
   > You can download this certificate from the SAP HANA server. The certificate `default.root.crt.pem` is typically installed in the following default location:
   >
   > `<installation_path>/<SID>/xs/controller_data/controller/ssl-pub/router/default.root.crt.pem`

   b. Switch to the organization and space in which SAP Medical Research Insights is installed and list all applications installed in that space by running the following commands:

   ```
   xs t -o <org name> -s <space name>
   xs a
   ```

   c. Display the information for the application `mri-pa-svc` by entering the following command:

   ```
   xs app mri-pa-svc
   ```

   d. In the details displayed, make a note of the value shown for `urls`.
2. Display all the environment variables set for the `mri-pa-svc` application so that you can identify what's set for the client ID and client secret.
   a. Run the following command to display all environment variables set:

   ```
   xs env mri-pa-svc
   ```

   b. From the JSON output that displays the variables set, navigate to the `"xsuaa"` element and make a note of the values shown for the `"clientid"` and `"clientsecret"` variables.
3. Using an HTTP client of your choice, request an `OAuth 2.0` authorization token that enables you to send an HTTP POST request. Do this using the following information:

Information in Request	Description or Information Source
Token name	Can be freely defined
Grant type	Password credentials
Access token URL	`https://<hostname>:3<instance number>32/uaa-security/oauth/token`

Information in Request	Description or Information Source
Username	Name of the user you use to access SAP Medical Research Insights
Password	Password for your SAP Medical Research Insights user
Client ID	The value noted for `"clientid"` in step 2
Client secret	The value noted for `"clientsecret"` in step 2

4. Still in your HTTP client, use the access token that you've generated to send a POST request, with your JSON schema as the body, to your instance.
   a. Locate your custom JSON schema that you tailored to match your data model.
   b. Open the JSON file in a suitable editor and insert a new first line with `{"schema":`. Also insert the corresponding closing `}` as the final line.
   c. Paste the contents of this JSON file into the area of your HTTP client that contains the body for the POST request.

      > → Tip
      >
      > For an example of the custom part of a JSON file, see Preparing Data to Expose to R [page 37].

   d. Enter the following information for the request header:

Information in Request	Description or Information Source
URL for POST	`https://<hostname>:<port>/sap/hc/mri/api/services/data/schema` You use the host name and port that you noted for `urls` for the `mri-pa-svc` application.
Content-Type	application/json

   e. When you've entered the body and header information, send the request.

      > i Note
      >
      > If, at any time, you want to check what JSON is currently uploaded to the server, you can also use the same end point above to retrieve the information with a GET request.

You've now uploaded your JSON schema to the SAP HANA server on which SAP Medical Research Insights is installed. Now that your JSON schema is in place, you can map the attributes of your data model to your FHIR-like JSON entities. You use your custom schema as a basis for this mapping. And the system uses this schema as a reference to form the JSON output as the response to API calls.

4.1.3 Mapping JSON Entities to Attributes

You map your JSON entities and their properties to the interaction attributes of your clinical data model. You do so by adding annotations to these attributes in your clinical data model configuration.

Context

You've created a custom JSON schema with FHIR-like entities and now want to map these entities to attributes in your clinical data model. To map your JSON entities to your clinical data model interaction attributes, you add an annotation to each of these attributes in the *Clinical Data Model Configuration* application.

> **i Note**
>
> The task outlined here assumes that you've already set up your clinical data model configuration. For more information about creating clinical data model configurations, see the *Clinical Data Model Configuration* section in the SAP Connected Health Platform Administration Guide.

Procedure

1. Go to the SAP Fiori launchpad for SAP Connected Health.

   The SAP Fiori launchpad is available at the following URL: `https://<WebServerHost>:3<SAPHANAinstance>30/go/chp-<space>`
2. Open the *Clinical Data Model Configuration* application.
3. Open a version of an existing configuration and edit the version.
4. For your basic data or each interaction that you want to allow to be used in a custom chart or analyzed externally, open the settings for the basic data or interaction under *Defined Interactions*.
5. For each attribute that you want to allow to be used in a custom chart or analyzed externally, open the settings for that attribute under *Basic Data* or *Saved Attributes*, and navigate to the *Annotations* field.
6. In the *Annotations* field, add an annotation in the following format:

   `data://<JSON entity>/<JSON property>`

   The entity and property names that you enter in your annotation must correspond with the entities and properties in your custom JSON schema.

   > **i Note**
   >
   > Entity and property names are case-sensitive. Each annotation that you enter must also be unique and configured just once.
7. Repeat steps 4–6 for all interactions and attributes that are to be available as data to be exposed or analyzed externally.

> **Note**
>
> Always add an annotation to the patient ID attribute. This is an important key that must be exposed.

> **Example**
>
> The following table provides some examples of mappings between JSON entities and properties and clinical data model configuration interactions and attributes:

JSON Entity	JSON Property	Interaction	Attribute	Attribute Annotation
Patient	identifier	Basic Data	Patient ID	`data://Patient/identifier`
Patient	gender	Basic Data	Gender	`data://Patient/gender`
Vitalstatus	vitalStatus	Vital Status	Vital Status	`data://Vitalstatus/vitalStatus`
Chemotherapy	opsCode	Chemotherapy	OPS Code	`data://Chemotherapy/opsCode`
Chemotherapy	protocol	Chemotherapy	Protocol	`data://Chemotherapy/protocol`

8. Validate, activate, and save your configuration. Make note of the configuration name and version as you need this information in the next step.
9. Use this clinical configuration version in any *Patient Analytics* configurations to be used by clinicians or researchers who want to use a custom chart or analyze data externally.
    a. Use the new clinical data model configuration version in a new or existing *Patient Analytics* configuration.

    > **→ Remember**
    >
    > If you want users to be able to expose data for external analysis from saved filters, ensure that you activate the *Make Data Accessible Externally* option for the configuration or configurations assigned to the users.

    For more information about assigning configurations, see the *Configuring Patient Analytics* section of the SAP Medical Research Insights Administration Guide.

    b. Assign this *Patient Analytics* configuration to the relevant organization or users in the *Configuration Assignment* application.

    For more information about assigning configurations, see the *Assigning Configurations to Users* section of the SAP Connected Health Platform Administration Guide.

4.1.4 Defining and Saving Filters for the Data Access API

As a clinician or researcher, you define the patient groups for which data is to be exposed from *Patient Analytics*.

Context

If you want to expose data from *Patient Analytics* to analyze it in external applications, you start by defining your patient set by applying filters and then saving your defined filters. You generate the access point based on a saved filter. The filters defined in your saved filter do not influence the type of data exposed. But they influence the patient groups for which that data is exposed. Your system administrator configures what data (interactions and attributes) can be exposed.

> **i** Note
>
> When data is loaded from an external application, the performance depends on the volume of data that has been exposed. The larger the volume of data, the longer the loading time. We therefore recommend that you keep performance in mind when applying and saving your filters for the data to be exposed.

Procedure

1. Go to the SAP Fiori launchpad for SAP Medical Research Insights.

   The SAP Fiori launchpad is available at the following URL: `https://<WebServerHost>:3<SAPHANAinstance>30/go/chp-<space>`
2. Open the *Patient Analytics* application.
3. Add filter cards and apply attributes to drill down to your patient group of interest.
4. To save your current filters as a saved filter, choose *Save*, enter a descriptive name, and save.

   You've now saved your current filters as a saved filter that can be used as the basis to generate the URL for the data access API.

4.1.5 Generating the Data Access API

As a clinician or researcher, you can expose data from *Patient Analytics* using the *Data Access API* option available for saved filters.

Context

You've created a saved filter in *Patient Analytics*, and you want to expose the data for patients that match those filters by generating a URL for the data access API. You want to use this API to consume and analyze data from external applications such as R.

Procedure

1. Go to the SAP Fiori launchpad for SAP Medical Research Insights.

   The SAP Fiori launchpad is available at the following URL: `https://<WebServerHost>:3<SAPHANAinstance>30/go/chp-<space>`
2. Open the *Patient Analytics* application.
3. Switch from the *Current Filters* to the *Saved Filters* tab.
4. Identify the saved filter for which you would like to expose the data for patients that match those filters and choose *(Make Data Accessible Externally)*.
5. On the *Data Access API* tab, choose *Copy URL* to copy the URL that's generated for use in external applications.
6. To test the data returned by the API, simply paste the URL into a new browser tab or window. Calling this API in your browser allows you to confirm that the data returned includes the information that you expect before you use the data in an external application.

   You've now completed the steps required to prepare the patient group and generate the API required to expose and access data from an external application.

4.1.6 Loading and Using the Data Exposed by the API

You can load data exposed by the SAP Medical Research Insights data access API from external applications.

Context

You've successfully generated a data access API from within *Patient Analytics*. You now want to load the data exposed by this API into an external application. The exact steps that you follow depend on the external application you use. The steps outlined here give a generic overview of the process.

When you load exposed data into an external application, you must have the following information:

Information Placeholder	Information Required
`<hostname>` `<port>`	The host name and port number for SAP Medical Research Insights **i Note** This port number must be the port that can be accessed from external applications, which is the `mri-pa-svc` port number. For information about how to identify this port, see step 1 of Uploading Your JSON Schema [page 12].
`<uid>` `<pwd>`	Credentials (user ID and password) to log on to SAP Medical Research Insights as a clinician
`<client_id>` `<client_secret>`	The client ID and client secret from environment variables set for the `mri-pa-svc` application. **→ Tip** For information about how to identify the client ID and client secret, see step 2 of Uploading Your JSON Schema [page 12].
`<patientGroupDefinition>`	The ID of the patient group definition from the API URL generated using the *Make Data Accessible Externally* > *Data Access API* option in *Patient Analytics*.
`<Entity>` **i Note** By default, the API exposes data for the `Patient` entity, which means you see this entity in the URL generated. You can simply replace this entity with any another entity that has been defined and exposed.	The names of the entities for which data has been exposed. Entity names are based on the custom JSON schema uploaded to the server and the corresponding annotations added in the *Clinical Data Model Configuration* application.

i Note

When data is loaded from an external application, the performance depends on the volume of data that has been exposed. The larger the volume of data, the longer the loading time. We therefore recommend that you keep performance in mind when applying and saving your filters for the data to be exposed.

Procedure

1. Use `https://<hostname>:3<instance number>32/uaa-security/oauth/token` to connect to the SAP Medical Research Insights server and get the relevant access token. Do this using the following information indicated in the table: `<hostname>`, `<port>`, `<uid>`, `<pwd>`, `<client_id>`, and `<client_secret>`.

2. Load the data exposed by the data access API by calling the API URI, which must contain the specific `<patientGroupDefinition>` and `<Entity>` from which you want to load data.

> **i Note**
>
> When you consume data exposed by the data access API, you can add query parameters to the URI to further filter the data. The operators currently supported area:
>
> - Logical operator: `OR`
> - Comparison operators: `eq` (equals), `ne` (not equals)
>
> > **Example**
> >
> > Imagine you want to retrieve basic patient data for females or non-smokers. So the query parameters you want to add are: `_filter=gender eq F or smoker ne y`
> >
> > The URI you access has the format:
> >
> > ```
> > https://<hostname>:<port>/sap/hc/mri/api/services/data/
> > <patientGroupDefinition>/Patient
> > ```
> >
> > The final URI that you call to filter your data is:
> >
> > ```
> > https://<hostname>:<port>/sap/hc/mri/api/services/data/
> > <patientGroupDefinition>/Patient?_filter=gender eq F or smoker ne y
> > ```

3. Repeat the API call for each entity (such as an interaction) that you want to load.
4. Convert the JSON result returned into whatever format you require for your external application.
5. Analyze the data as required in your external application.

4.2 Tutorial: Exposing Data from Patient Analytics for Analysis in R

Use this example to understand how you could expose data for analysis in R, and then consume and analyze that data in R.

Before you start, in addition to the following prerequisites specific to R, ensure that you've met all the technical prerequisites outlined in Technical Prerequisites [page 5].

Required to Use R as Data Analysis Extension to SAP Medical Research Insights

- You've installed RStudio on your local machine (on which you want to analyze SAP Medical Research Insights data externally).
- You're using R version 3.4.4 or higher.
- You've installed the following R packages:

- `httr`
- `jsonlite`
- `dplyr`
- `tidyverse`
- `sjPlot` (if you want to visualize effects plots of regression models, for example)

> **i Note**
>
> These packages are publicly available in the R package repository hosted by the Comprehensive R Archive Network (CRAN).

Example Use Case

Imagine that as a researcher, you use *Patient Analytics* to filter, group, and analyze patient data to verify a research hypothesis. This hypothesis is on the efficacy of chemotherapy treatment for lung cancer based on gender.

You want to analyze the data further by using the logistic regression model in R. So you save the filters you used to build your patient group and generate the data access API from that saved filter. The data that you need is basic data for the patient, such as patient ID, gender, and vital status. So let's look at the steps to expose the data from *Patient Analytics* and to load the data within R.

4.2.1 Preparing Data to Expose to R

As system administrator you complete the steps required to enable your clinicians to expose data from *Patient Analytics* to R.

Context

These preparatory steps involve:

- Creating a JSON schema to reflect your data model
- Uploading this schema to the server
- Mapping your interaction types and attributes to the JSON schema entities in configurations assigned to you
- Enabling the option for users to make data accessible externally

Procedure

1. Create your custom JSON schema. For more information, see Creating Your Custom JSON Schema [page 10].

The data to be exposed in this example includes the patient identifier and gender of the patient, which belong to the basic patient data or the patient entity. These entities are already included in the standard FHIR JSON schema. But you still have to add your vital status entity to your custom JSON. This part of your custom JSON could look as follows, for example:

Sample Code

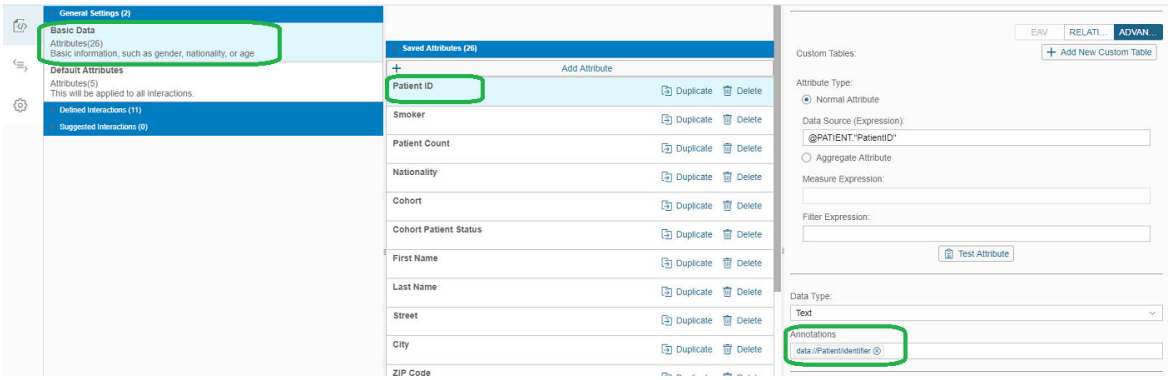
```
          "Vitalstatus": {
           "description": "Measurements and simple assertions made about a
patient, device or other subject.",
           "properties": {
             "resourceType": {
               "description": "This is a Vital status resource",
               "const": "Vitalstatus"
           },
           "id": {
              "description": "The logical id of the resource, as used in
the URL for the resource. Once assigned, this value never changes.",
               "$ref": "#/definitions/id"
           },
            "identifier": {
               "description": "A unique identifier assigned to this
observation.",
               "items": {
               "$ref": "#/definitions/Identifier"
               },
               "type": "array"
           },
            "startDateTime": {
               "description": "The time or time-period the observed value
is asserted as being true. For biological subjects - e.g. human patients -
this is usually called the \"physiologically relevant time\". This is
usually either the time of the procedure or of specimen collection, but
very often the source of the date/time is not known, only the date/time
itself.",
               "pattern": "^([0-9]([0-9]([0-9][1-9]|[1-9]0)|[1-9]00)|
[1-9]000)(-(0[1-9]|1[0-2])(-(0[1-9]|[1-2][0-9]|3[0-1])(T([01][0-9]|2[0-3]):
[0-5][0-9]:([0-5][0-9]|60)(\\.[0-9]+)?(Z|(\\+|-)((0[0-9]|1[0-3]):[0-5]
[0-9]|14:00)))?)?)?$",
               "type": "array"
           },
             "endDateTime": {
               "description": "The time or time-period the observed value
is asserted as being true. For biological subjects - e.g. human patients -
this is usually called the \"physiologically relevant time\". This is
usually either the time of the procedure or of specimen collection, but
very often the source of the date/time is not known, only the date/time
itself.",
               "pattern": "^([0-9]([0-9]([0-9][1-9]|[1-9]0)|[1-9]00)|
[1-9]000)(-(0[1-9]|1[0-2])(-(0[1-9]|[1-2][0-9]|3[0-1])(T([01][0-9]|2[0-3]):
[0-5][0-9]:([0-5][0-9]|60)(\\.[0-9]+)?(Z|(\\+|-)((0[0-9]|1[0-3]):[0-5]
[0-9]|14:00)))?)?)?$",
               "type": "array"
             },
             "ageAtLastContact": {
               "description": "sampleStatus",
               "$ref": "#/definitions/Reference",
                "type": "array"
             },
               "vitalStatus": {
               "description": "Vital Status",
               "$ref": "#/definitions/Reference",
               "type": "array"
               }
             },
             "additionalProperties": false,
```

```
            "required": [
              "code",
              "resourceType"
            ]
        }
```

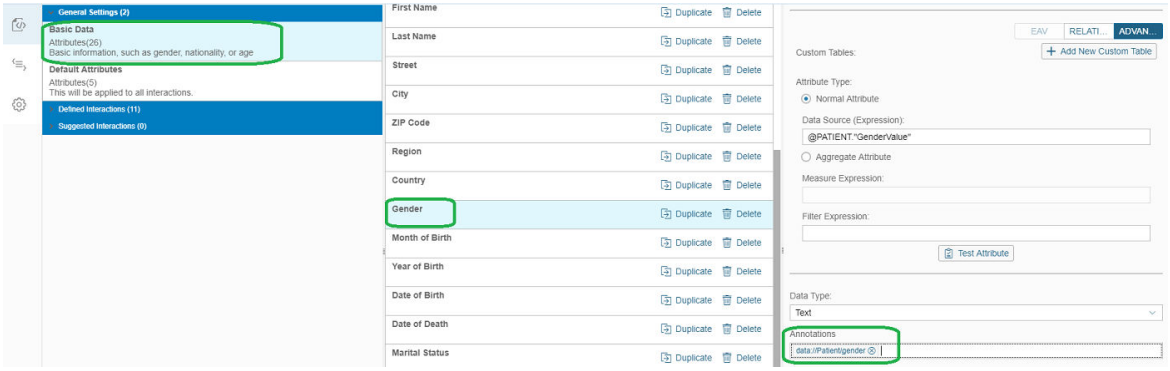
2. Upload your JSON schema. For more information, see Uploading Your JSON Schema [page 12].

   Again, in this example, the data to be exposed are the patient ID, gender, and vital status. So you locate these attributes in your clinical data model configuration and add annotations to them as shown in these examples:

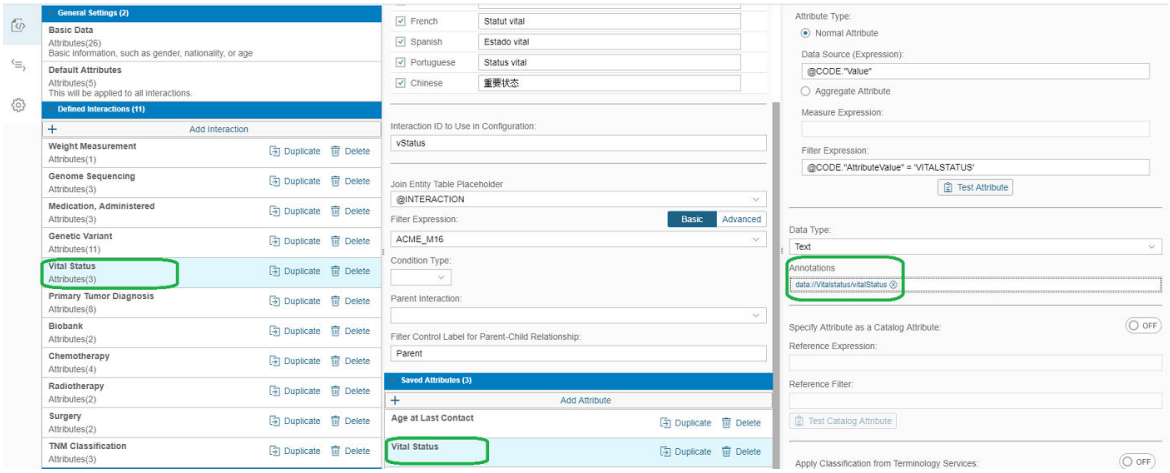
   Example of the annotation for the patient ID:



   Example of the annotation for the patient gender:



Example of the annotation for the vital status:



3. Map your JSON entities to interaction types and attributes in your clinical data model configuration. And use this configuration as the basis for the *Patient Analytics* configuration assigned to users. For more information, see Mapping JSON Entities to Attributes [page 15].

> → Remember
>
> If you want users to be able to expose data for external analysis from saved filters, ensure that you activate the *Make Data Accessible Externally* option for the configuration or configurations assigned to the users.

Results

When these steps are completed on the administration side, clinicians can expose data from *Patient Analytics*.

4.2.2 Exposing Data from Patient Analytics for Analysis in R

As a clinician or researcher, you can expose data from *Patient Analytics* to allow you to analyze it in external applications.

Context

Your system administrator has completed the steps required on the administrator side to allow you to expose data from *Patient Analytics*. These steps involve:

- Creating a JSON schema to reflect your data model
- Uploading this schema to the server
- Mapping your interaction types and attributes to the JSON schema entities in configurations assigned to you

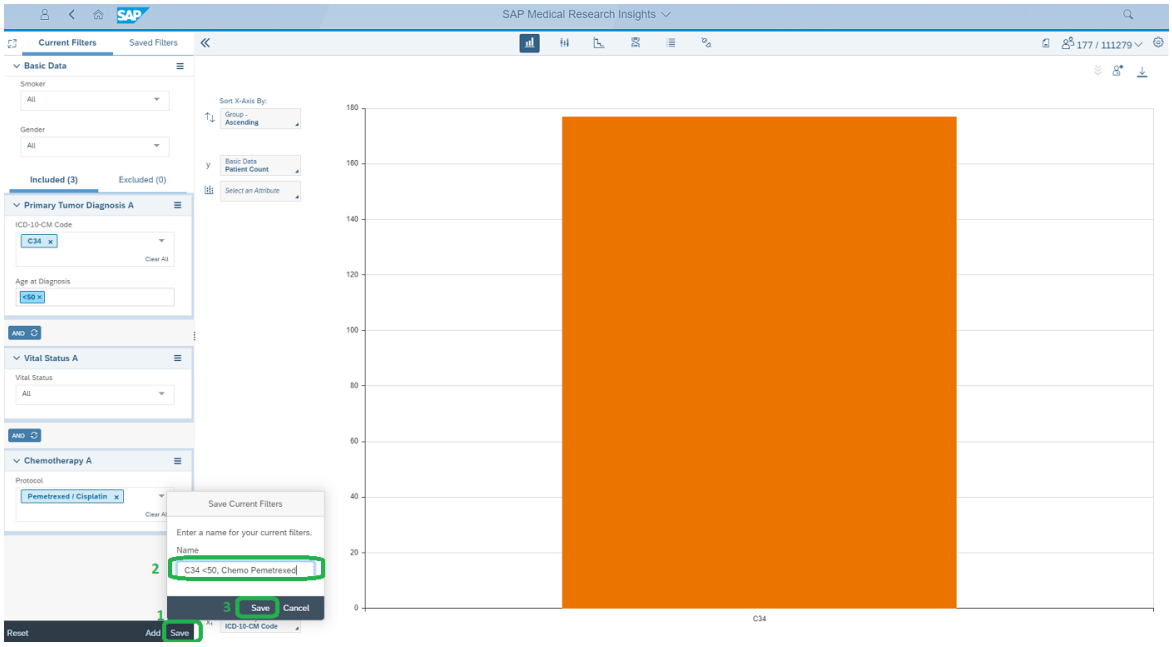
- Enabling the option for you to make your data accessible externally

Completing these steps means that the entities and properties that you want to expose are configured on the administration side. Once these steps are completed on the administration side, you can prepare to expose your data from *Patient Analytics*.

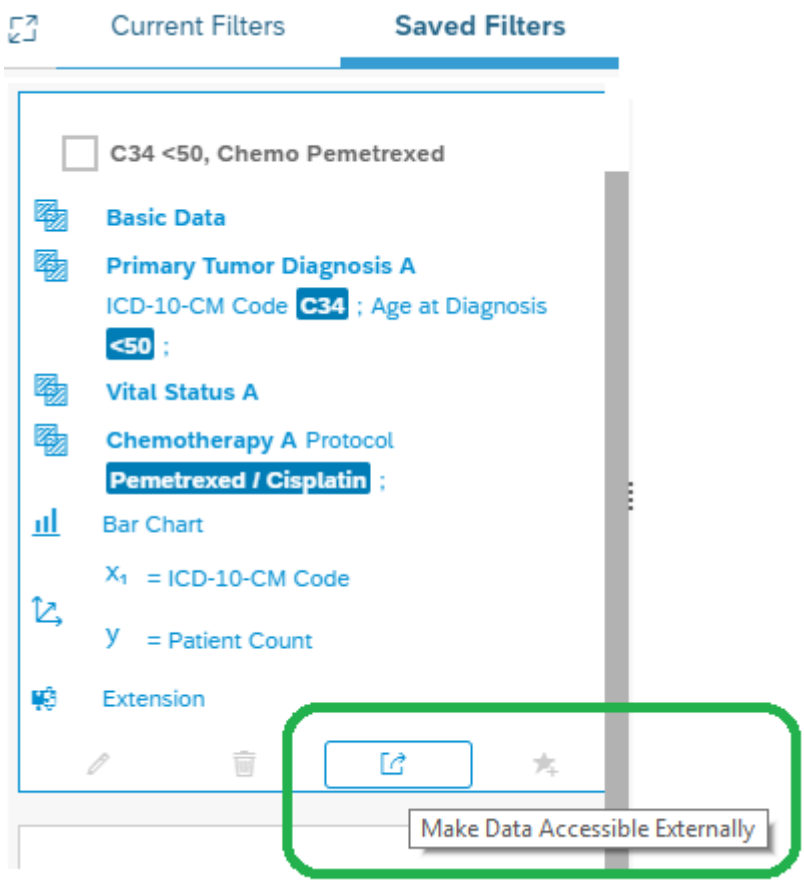
Procedure

1. Go to the SAP Fiori launchpad for SAP Medical Research Insights.

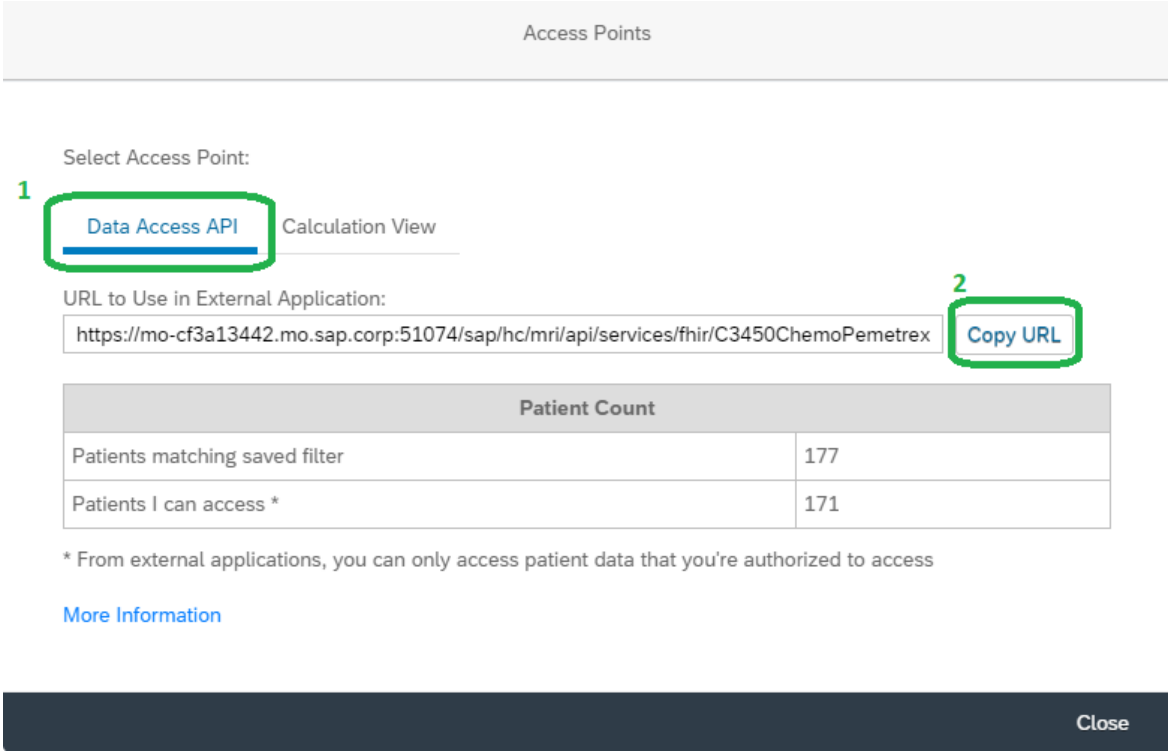
   The SAP Fiori launchpad is available at the following URL: `https://<WebServerHost>:3<SAPHANAinstance>30/go/chp-<space>`
2. Open the *Patient Analytics* application.
3. Add filter cards and apply attributes to filter to your patient group of interest.
4. Save your current filters to generate a saved filter:



5. Switch to the *Saved Filters* tab and choose the save filter you just created.
6. Choose *(Make Data Accessible Externally)*:



7. On the *Data Access API* tab, copy the URL that's generated for use in external applications:



> **i Note**
>
> From external applications such as R, you can only access patient data that you're authorized to access. These patients are the same patients that you are authorized to access from the patient list, for example. The number of patients that you can access is indicated in the patient count information provided on this screen.

You've now completed the steps required to prepare the patient group and generate the API URL required to access data from R.

4.2.3 Loading Data Exposed for Analysis into R

When you have exposed data from *Patient Analytics* using the data access API, you use the API URL generated to load your data into R.

Context

When you load data from *Patient Analytics* into R, you must have the following information:

Information Placeholder	Information Required
`<hostname>` `<port>`	The host name and port number for SAP Medical Research Insights **i Note** This port number must be the port that can be accessed from external applications, which is the `mri-pa-svc` port number. For information about how to identify this port, see step 1 of Uploading Your JSON Schema [page 12].
`<uid>` `<pwd>`	Credentials (user ID and password) to log on to SAP Medical Research Insights as a clinician
`<client_id>` `<client_secret>`	The client ID and client secret from environment variables set for the `mri-pa-svc` application. **→ Tip** For information about how to identify the client ID and client secret, see step 2 of Uploading Your JSON Schema [page 12].
`<patientGroupDefinition>`	The ID of the patient group definition from the API URL generated using the *Make Data Accessible Externally* > *Data Access API* option in *Patient Analytics*.
`<Entity>` **i Note** By default, the API exposes data for the `Patient` entity, which means you see this entity in the URL generated. You can simply replace this entity with any another entity that has been defined and exposed.	The names of the entities for which data has been exposed. Entity names are based on the custom JSON schema uploaded to the server and the corresponding annotations added in the *Clinical Data Model Configuration* application.

Procedure

1. Open RStudio and copy the following template into a new R script. The code in this template essentially handles the following:
   - Connects to the SAP Medical Research Insights server
   - Gets the relevant access token
   - Loads the data exposed by the data access API. Data can be exposed for the patient entity and any other entities. You retrieve data separately for each entity.
   - Converts the JSON result returned into an R data frame
   - Unnests data list columns to make each element of the list its own row

   Sample Code

   ```
   library(httr)
   library(jsonlite)
   ```

```
library(dplyr)
library(tidyverse)
url <- "https://<hostname>:30032/uaa-security/oauth/token"
uid <- "<uid>"
pwd <- "<pwd>"
clientid <- "<client_id>"
clientsecret <- "<client_secret>"
httr::set_config(config(ssl_verifypeer = 0L, ssl_verifyhost = 0L))
res <- POST(url,
            body = list(grant_type = "password", username = uid, password
= pwd),
            authenticate(clientid, clientsecret),
            encode = "form")
token <- paste("Bearer", httr::content(res)$access_token)
api_patient="https://<hostname>:<port>/sap/hc/mri/api/services/data/
<patientGroupDefinition>/Patient"
api_1="https://<hostname>:<port>/sap/hc/mri/api/services/data/
<patientGroupDefinition>/<Entity>"
raw_patient <- httr::GET(api_patient, httr::add_headers(Authorization =
token))
raw_data1 <- httr::GET(api_1, httr::add_headers(Authorization = token))
c_patient <-httr::content(raw_patient, as="text", encoding = "UTF-8")
c_data1 <-httr::content(raw_data1, as="text", encoding = "UTF-8")
patient <- fromJSON(c_patient)
data1 <- fromJSON(c_data1)
patient <- as.data.frame(patient$entry)
data1 <- as.data.frame(data1$entry)
patient <- patient %>% unnest()
data1 <- data1 %>% unnest()
```

2. Replace all placeholders with the relevant values.
3. Tailor the code to reflect from which entities you want to load data that's been exposed, and add the code to run the analysis. In this example, the entities accessed are `Patient` and `Vitalstatus`:

Sample Code

```
library(httr)
library(jsonlite)
library(dplyr)
library(tidyverse)
url <- "https://MyHostName:30032/uaa-security/oauth/token"
uid <- "MyUserName"
pwd <- "MyPassword"
clientid <- "MyClientID"
clientsecret <- "MyClientSecret"
httr::set_config(config(ssl_verifypeer = 0L, ssl_verifyhost = 0L))
res <- POST(url,
            body = list(grant_type = "password", username = uid, password
= pwd),
            authenticate(clientid, clientsecret),
            encode = "form")
token <- paste("Bearer", httr::content(res)$access_token)
#Port number must be mri-pa-svc port number
api_patient="https://<hostname>:<port>/sap/hc/mri/api/services/data/
MyPatientGroupDefinition/Patient"
api_vitalstatus="https://<hostname>:<port>/sap/hc/mri/api/services/data/
MyPatientGroupDefinition/Vitalstatus"
raw_patient <- httr::GET(api_patient, httr::add_headers(Authorization =
token))
raw_vitalstatus <- httr::GET(api_vitalstatus,
httr::add_headers(Authorization = token))
c_patient <-httr::content(raw_patient, as="text", encoding = "UTF-8")
c_vitalstatus <-httr::content(raw_vitalstatus, as="text", encoding =
"UTF-8")
patient <- fromJSON(c_patient)
```

```
vitalstatus <- fromJSON(c_vitalstatus)
#Get the content from entry node
patient <- as.data.frame(patient$entry)
vitalstatus <- as.data.frame(vitalstatus$entry)
#Unnest the data as it is likely to have multiple values
patient <- patient %>% unnest()
vitalstatus <- vitalstatus %>% unnest()
#Join entity for this example is "identifier", which is PatientID. Update
accordingly as per your configuration
#Perform left join for patient and vital status
df <-merge(patient, vitalstatus,  by="identifier", all.x=TRUE)
#Set missing value as NA
df[is.na(df)] <- 'NA'
#Set vital status to 1 (alive) or 0 (deceased)
df$vstatus <- as.numeric(ifelse( df$vitalStatus == 'Alive', '1', '0'))
#Dependant var: vital status
#Independant var: gender
final_data <- df[,c("_gender", "vstatus")]
#set appropriate column names by renaming the existing names
names(final_data) <- c("gender", "vital_status")
#set gender as factor
final_data$gender <-as.factor(final_data$gender)
summary(final_data)
library(sjPlot)
model <-glm(vital_status ~ gender, data=final_data, family="binomial")
#display model
model
```

5 Security Considerations

From a security perspective, there are aspects you should be aware of when exposing data or creating or working with extensions for *Patient Analytics*.

Relevant for **data consumers**:

- All data access and retrieval should be done using a secure communication protocol (HTTPS).
- A valid user name and password for SAP Medical Research Insights are required to consume data externally.
- OAuth 2.0 access tokens must be generated to authenticate API requests made using your user credentials.
- When patient data is exposed for analysis, the security settings of SAP Medical Research Insights and SAP Connected Health platform apply. This means that patient data is exposed only if the person exposing it is authorized to view that data, and this data is assigned to the same organizational unit as the person exposing it.
- When you expose data using the data access API or in a calculation view, only guarded patient data is exposed and can be accessed from external applications.
- When you access and consume data externally from a cloud application, such as SAP Analytics Cloud, this data is not stored externally. You connect to the SAP HANA server from your browser and simply retrieve the data for display in SAP Analytics Cloud.
- Similarly, when you access and consume data externally using the data access API from analysis applications such as R, this data is retrieved but is not stored externally.

Relevant for **extension developers**:

- All data access and retrieval should be done using a secure communication protocol (HTTPS).
- When you create extensions, you are responsible for ensuring you take all security measures required, for example, to prevent malicious code being injected into your custom code.
- If users are to consume data from a calculation view in an external tool such as SAP Cloud Analytics, you must configure SAML settings to authenticate such users accessing data from external applications.

Important Disclaimers and Legal Information

Hyperlinks

Some links are classified by an icon and/or a mouseover text. These links provide additional information.
About the icons:

- Links with the icon  : You are entering a Web site that is not hosted by SAP. By using such links, you agree (unless expressly stated otherwise in your agreements with SAP) to this:
  - The content of the linked-to site is not SAP documentation. You may not infer any product claims against SAP based on this information.
  - SAP does not agree or disagree with the content on the linked-to site, nor does SAP warrant the availability and correctness. SAP shall not be liable for any damages caused by the use of such content unless damages have been caused by SAP's gross negligence or willful misconduct.
- Links with the icon : You are leaving the documentation for that particular SAP product or service and are entering a SAP-hosted Web site. By using such links, you agree that (unless expressly stated otherwise in your agreements with SAP) you may not infer any product claims against SAP based on this information.

Beta and Other Experimental Features

Experimental features are not part of the officially delivered scope that SAP guarantees for future releases. This means that experimental features may be changed by SAP at any time for any reason without notice. Experimental features are not for productive use. You may not demonstrate, test, examine, evaluate or otherwise use the experimental features in a live operating environment or with data that has not been sufficiently backed up.
The purpose of experimental features is to get feedback early on, allowing customers and partners to influence the future product accordingly. By providing your feedback (e.g. in the SAP Community), you accept that intellectual property rights of the contributions or derivative works shall remain the exclusive property of SAP.

Example Code

Any software coding and/or code snippets are examples. They are not for productive use. The example code is only intended to better explain and visualize the syntax and phrasing rules. SAP does not warrant the correctness and completeness of the example code. SAP shall not be liable for errors or damages caused by the use of example code unless damages have been caused by SAP's gross negligence or willful misconduct.

Gender-Related Language

We try not to use gender-specific word forms and formulations. As appropriate for context and readability, SAP may use masculine word forms to refer to all genders.

www.sap.com/contactsap

© 2019 SAP SE or an SAP affiliate company. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP SE or an SAP affiliate company. The information contained herein may be changed without prior notice.

Some software products marketed by SAP SE and its distributors contain proprietary software components of other software vendors. National product specifications may vary.

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

SAP and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP SE (or an SAP affiliate company) in Germany and other countries. All other product and service names mentioned are the trademarks of their respective companies.

Please see https://www.sap.com/about/legal/trademark.html for additional trademark information and notices.

THE BEST RUN SAP