Business Rules Capability for the Cloud Foundry Environment

Generated on: 2021-07-25 06:41:09 GMT+0000

Business Rules Capability Within the SAP Workflow Management Service | Cloud

PUBLIC

Original content: https://help.sap.com/viewer/0e4dd38c4e204f47b1ffd09e5684537b/Cloud/en-US

Warning

What Is Business Rules Capability?

Digitize and automate decision making with business rules.

The business rules capability within the SAP Workflow Management service lets you digitize and automate decision making. It encapsulates dynamic decision logic from application logic. It also provides web-based tools to model business vocabulary and decision interfaces to integrate with SaaS applications and business services.

Environment

This capability runs in the Cloud Foundry and Neo environments.

Features

Manage rules in a web-based application

Manage the lifecycle of business rule projects across business systems using a central repository. Deploy rule services to multiple runtime environments like SAP BTP rule runtime, SAP S/4HANA Cloud, SAP S/4HANA or SAP HANA.

Embed business rules

Embed business rules like decision tables and text rules in SaaS application user interface using SAP UI5 Rule Builder Control. Offers an in-app experience for business users to configure dynamic business policies without affecting the user experience.

Scope and Limitations

For information on technical limits, see Conventions, Restrictions, and Limits.

Regional Availability

Get an overview on the availability of business rules capability according to region, infrastructure provider, and release status: Discovery Center Service Catalog.

Trial Scope

Business Rules Capability is available for trial use. A trial account lets you try out SAP BTP for free and is open to everyone. Trial accounts are intended for personal exploration, and not for productive use or team development. They allow restricted use of the platform resources and services. The trial period varies depending on the environment.

To activate your trial account, go to Welcome to SAP BTP Trial.

There are no restrictions for the trial landscape. The lite plan is supported for the trial users.

> **Note**
> See also the following information: Trial Accounts.

What's New for Business Rules Capability

2020

Technical Component	Capability	Environment	Title	Description	Type	Available as of

Technical Component	Capability	Environment	Title	Description	Type	Available as of
Business Rules Capability	Extension Suite - Digital Process Automation	Cloud Foundry Neo	Business Rules	The SAP Business Rules Service is now available as a capability of SAP Workflow Management. As part of this change, the capability is renamed to "business rules capability within the SAP Workflow Management service" or "business rules" in short.	Changed	2021-07-21
Business Rules Capability	Extension Suite - Digital Process Automation	Cloud Foundry Neo	System for Value Help	You can now configure the system to retrieve the value help for each data object. See Model Data Objects.	New	2021-07-21
Business Rules Capability	Extension Suite - Digital Process Automation	Cloud Foundry	Boosters	The booster will now create additional service specific destinations based on the service instance to access the launchpad. See Initial Setup.	Changed	2021-06-08
Business Rules Capability	Extension Suite - Digital Process Automation	Cloud Foundry	Region - Asia Pacific (Seoul)	Business rules capability is now available on Asia Pacific (Seoul) (cf-ap12) as new region running on Amazon Web Services for use with enterprise accounts. See What Is Business Rules Capability?	Announcement	2021-03-31
Business Rules Capability	Extension Suite - Digital Process Automation	Neo Cloud Foundry	Revision-Based APIs	Using revision-based APIs, you can now specify the revision of a project to: • Retrieve the content or the entities of the project • Export or import the project See *Rule Authoring API* in SAP Business Rules Service APIs.	New	2021-03-05
Business Rules Capability	Extension Suite - Digital Process Automation	Cloud Foundry	Region - Canada (Montreal)	Business rules capability is now available on Canada (Montreal)(cf-ca10) as new region running on Amazon Web Services for use with enterprise accounts. See What Is Business Rules Capability?	Announcement	2021-02-18
Business Rules Capability	Extension Suite - Digital Process Automation	Neo Cloud Foundry	Advanced Functions	You can now perform currency conversion using the advanced function **CONVERTAMOUNT**. See Functions.	New	2021-02-18
Business Rules Capability	Extension Suite - Digital Process Automation	Neo Cloud Foundry	Updated Branding	Documentation and UI labels updated to reflect updated branding.	Changed	2021-02-10
Business Rules Capability	Extension Suite - Digital Process Automation	Neo Cloud Foundry	Copy a Rule	You can now copy a rule instead of creating a new rule. See Copy a Rule.	New	2021-02-10

Technical Component	Capability	Environment	Title	Description	Type	Available as of
Business Rules Capability	Extension Suite - Digital Process Automation	Neo Cloud Foundry	Vocabulary Rules	You can now use a decision table with **All match** hit policy as a vocabulary rule.	New	2021-02-10
Business Rules Capability	Extension Suite - Digital Process Automation	Neo Cloud Foundry	Advanced Functions	The following functions are now supported on cloud runtime: **ISNULL** and **ISNOTNULL**. See Functions.	New	2021-02-10
Business Rules Capability	Extension Suite - Digital Process Automation	Neo Cloud Foundry	Reference Currency	For ABAP runtime, you can now configure **Reference Currency** to perform currency conversions during the rule execution. See Create a Project.	New	2021-02-10

Related Information

2020 What's New for Business Rules Capability (Archive)

2020 What's New for Business Rules Capability (Archive)

2020

Technical Component	Capability	Environment	Title	Description	Type	Available as of
Business Rules Capability	Extension Suite - Digital Process Automation	Neo Cloud Foundry	Performance improvement	The performance of rule authoring APIs is now improved. [Customer Feedback]	Announcement	2020-12-24
Business Rules Capability	Extension Suite - Digital Process Automation	Neo Cloud Foundry	Feature enhancement	Validation of rules in Expression Language 2.0 is now improved.	Changed	2020-12-24
Business Rules Capability	Extension Suite - Digital Process Automation	Neo Cloud Foundry	Loop functions	You can use the loop function **FOREACH** to perform execute operations in a loop. See Model a Text Rule	New	2020-12-24
Business Rules Capability	Extension Suite - Digital Process Automation	Neo Cloud Foundry	New business data type	Amount business data type is now available in Expression Language 2.0. See Expression Language 2.0.	New	2020-12-24
Business Rules Capability	Extension Suite - Digital Process Automation	Neo Cloud Foundry	Timestamp format for rule service result	You can now configure the timestamp format for the rule service result in the cloud runtime. See Model a Rule Service.	New	2020-12-24

Technical Component	Capability	Environment	Title	Description	Type	Available as of
Business Rules Capability	Extension Suite - Digital Process Automation	Neo Cloud Foundry	Labels for decision table column headers	You can now provide labels for condition expressions in decision table column headers. See Model a Rule in Decision Table.	New	2020-12-24
Business Rules Capability	Extension Suite - Digital Process Automation	Cloud Foundry	Boosters	You can now setup business rules capability automatically using a booster. See Initial Setup.	New	2020-12-03
Business Rules Capability	Extension Suite - Digital Process Automation	Cloud Foundry	New trial region	Business Rules is now available on Singapore (cf-ap21) as a new region running on Microsoft Azure for use with trial accounts. See Initial Setup.	Announcement	2020-10-22
Business Rules Capability	Extension Suite - Digital Process Automation	Neo Cloud Foundry	Performance of Rule Service Invocation	The performance of the rule service invocation is now improved. [Customer Feedback]	Announcement	2020-10-22
Business Rules Capability	Extension Suite - Digital Process Automation	Cloud Foundry	Authentication type for destinations	You can now create destinations for managed systems using OAuth client credentials authentication. See Configure a System.	New	2020-10-01
Business Rules Capability	Extension Suite - Digital Process Automation	Neo Cloud Foundry	Decommission of version v1 of rule execution APIs	Version v1 of rule execution APIs will be decommissioned by the end of September 2020. You can migrate from version v1 to version v2 of rule execution APIs. See Migrate Business Rules Execution APIs to the New Version.	Announcement	2020-08-07
Business Rules Capability	Extension Suite - Digital Process Automation	Neo Cloud Foundry	UI enhancement	In the select and aggregate functions, you can now use a data object of type **Element** as a filter condition in the **Where** field. See Model a Rule Expression in Expression Language 2.0.	New	2020-08-07
Business Rules Capability	Extension Suite - Digital Process Automation	Neo Cloud Foundry	Updates	Minor updates and bug fixes.	New	2020-07-02
Business Rules Capability	Extension Suite - Digital Process Automation	Neo Cloud Foundry	UI enhancement	In the **Manage Rule Projects** application, you can now define the filters for displaying the list of projects or project entities using the ▽ *filter icon* .	Changed	2020-06-18

Technical Component	Capability	Environment	Title	Description	Type	Available as of
Business Rules Capability	Extension Suite - Digital Process Automation	Neo Cloud Foundry	Updates	Minor updates and bug fixes	New	2020-05-14
Business Rules Capability	Extension Suite - Digital Process Automation	Neo Cloud Foundry	Audit logs	You can now view the audit logs for the Business Rules Capability events. See • Audit Logs for Business Rules in the Cloud Foundry Environment • Audit Logs for Business Rules in the Neo Environment	New	2020-05-07
Business Rules Capability	Extension Suite - Digital Process Automation	Neo Cloud Foundry	Updates	Minor updates and bug fixes	New	2020-04-16
Business Rules Capability	Extension Suite - Digital Process Automation	Cloud Foundry	New enterprise regions	Business rules capability is now available on US East (VA)(cf-us21), Singapore (cf-ap21), and Japan (Tokyo)(cf-jp20) as new regions running on Microsoft Azure for use with enterprise accounts.	Announcement	2020-04-02
Business Rules Capability	Extension Suite - Digital Process Automation	Neo Cloud Foundry	Pagination in Business Rules APIs	Using client-driven pagination in business rules APIs, you can retrieve a defined number of projects. See SAP Business Rules Capability.	New	2020-03-06
Business Rules Capability	Extension Suite - Digital Process Automation	Neo Cloud Foundry	Pagination in Business Rules APIs	Using client-driven pagination in business rules service APIs, you can retrieve a defined number of entities like data object, rule, rule service, and ruleset. See SAP Business Rules Capability.	New	2020-02-20
Business Rules Capability	Extension Suite - Digital Process Automation	Neo Cloud Foundry	Select functions	**First** function is available under **Select Functions**. See Functions in Expression Language 2.0.	New	2020-02-20
Business Rules Capabilitye	Extension Suite - Digital Process Automation	Neo Cloud Foundry	Advanced functions	The following functions are available under **Advanced Functions**: Is initial, Is not initial. See Functions in Expression Language 2.0.	New	2020-02-20
Business Rules Capability	Extension Suite - Digital Process Automation	Neo Cloud Foundry	Text rule operations	You can model a text rule to perform append operations on data objects in expression language 2.0. See Model a Text Rule.	New	2020-02-20

Technical Component	Capability	Environment	Title	Description	Type	Available as of
Business Rules Capability	Extension Suite - Digital Process Automation	Neo Cloud Foundry	Advanced functions	The following functions are available under **Advanced Functions**: Is null, Is not null. See Functions in Expression Language 2.0.	New	2020-02-06
Business Rules Capability	Extension Suite - Digital Process Automation	Neo Cloud Foundry	Time and duration functions	The following functions are available under **Time and Duration Functions**: Second, Minute, Hour, Month, Year, Day of week, Day of month, Day of year. See Functions in Expression Language 2.0.	New	2020-02-06
Business Rules Capability	Extension Suite - Digital Process Automation	Neo Cloud Foundry	Free flow typing	Using free flow typing, you can edit or delete tokens in a rule expression. See Autosuggest List in Expression Language 2.0.	New	2020-02-06

Concepts

Business Rules Capability enables a cloud application developer to embed decisions into cloud extension applications and workflow applications.

A developer can expose the decision logic to different business users and knowledge experts, enabling them to directly author it. The developer at the customer site creates an application, which can include other services too.

In the customer subaccount, the developer can enable business rules to use the service in their application. The developer can also consume the REST-based and OData based APIs to define the rules for the application. The business key users can access the business logic via a custom fiori application or any other custom UI application.

The service has the following capabilities:

- **Create projects**

  Projects hold the business rules entities such as data objects, rules, rulesets, and rule services.

- **Add rule services to the projects to define the interface**

  Rule services are interfaces or end points that enables an application to invoke a decision logic.

- **Add business logic by modeling rules**

  Rules define a business logic that, once evaluated against live data, leads to a decision.

- **Configure the ruleset to link the business logic to the rule service**

  Ruleset serve as an entry point for rule processing, and links a rule service to a collection of rules.

  .

Conventions, Restrictions, and Limits

The following conventions, restrictions, and limits apply to Business Rules Capability. Considering this information during development, helps you to achieve an optimal use of business rules capability.

ℹ Note

Limits are, to the extent possible, subject to change.

Execution Limits

Area	Limit	Value for Basic/Standard Plan (Enterprise Account)		Value for Lite Plan (Trial Account)	More Information
		Rule Execution APIs	Rule Authoring APIs		
API	Request rate limit	150 requests per second per tenant	120 requests per second per tenant	30 requests per second per tenant	• Using Business Rules Capability APIs • Includes requests triggered from user interfaces delivered by SAP.
	Request body size	• 100 KB for execution APIs • 5 MB for JSON payload of project import API	400 KB		n.a

Area	Limit	Value for Basic/Standard Plan (Enterprise Account)		Value for Lite Plan (Trial Account)	More Information
		Rule Execution APIs	Rule Authoring APIs		
Rule service deployment	Size of the rule service deployment content	5 MB per rule service deployment			Rule service deployment content size is calculated at the time of the deployment based on the number and size of the following entities: • The input and result data object assigned to the rule service vocabulary. • The rulesets associated with the rule service. • The data objects and rules included in the ruleset vocabulary. • The rules that are assigned to the rulesets of the rule service.

Restrictions

Maximum number of versions of a project is restricted to 10.

Usage Scenarios

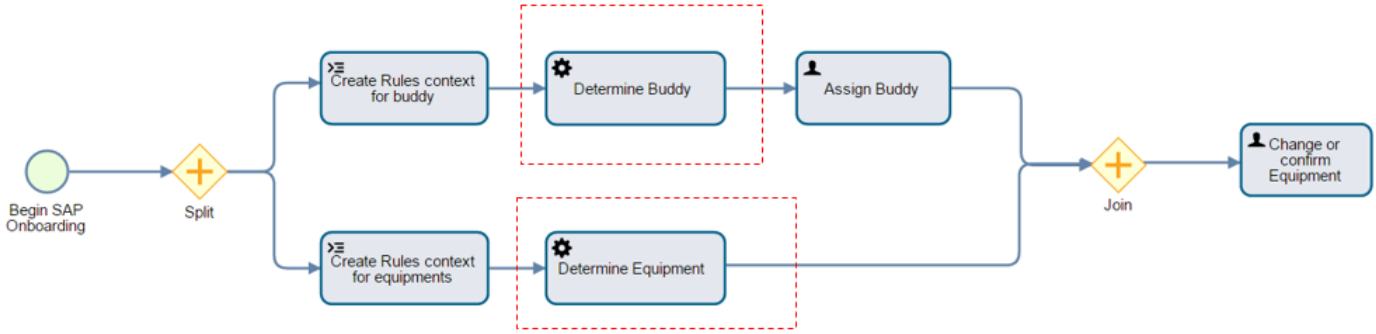
In the usage scenarios, you can find the use cases for the business rules capability.

Extend SAP Cloud Solutions

Developers can securely develop loosely coupled extension applications, implementing additional policies on top of the existing SAP cloud solution. The usage scenario consists of the following use cases:

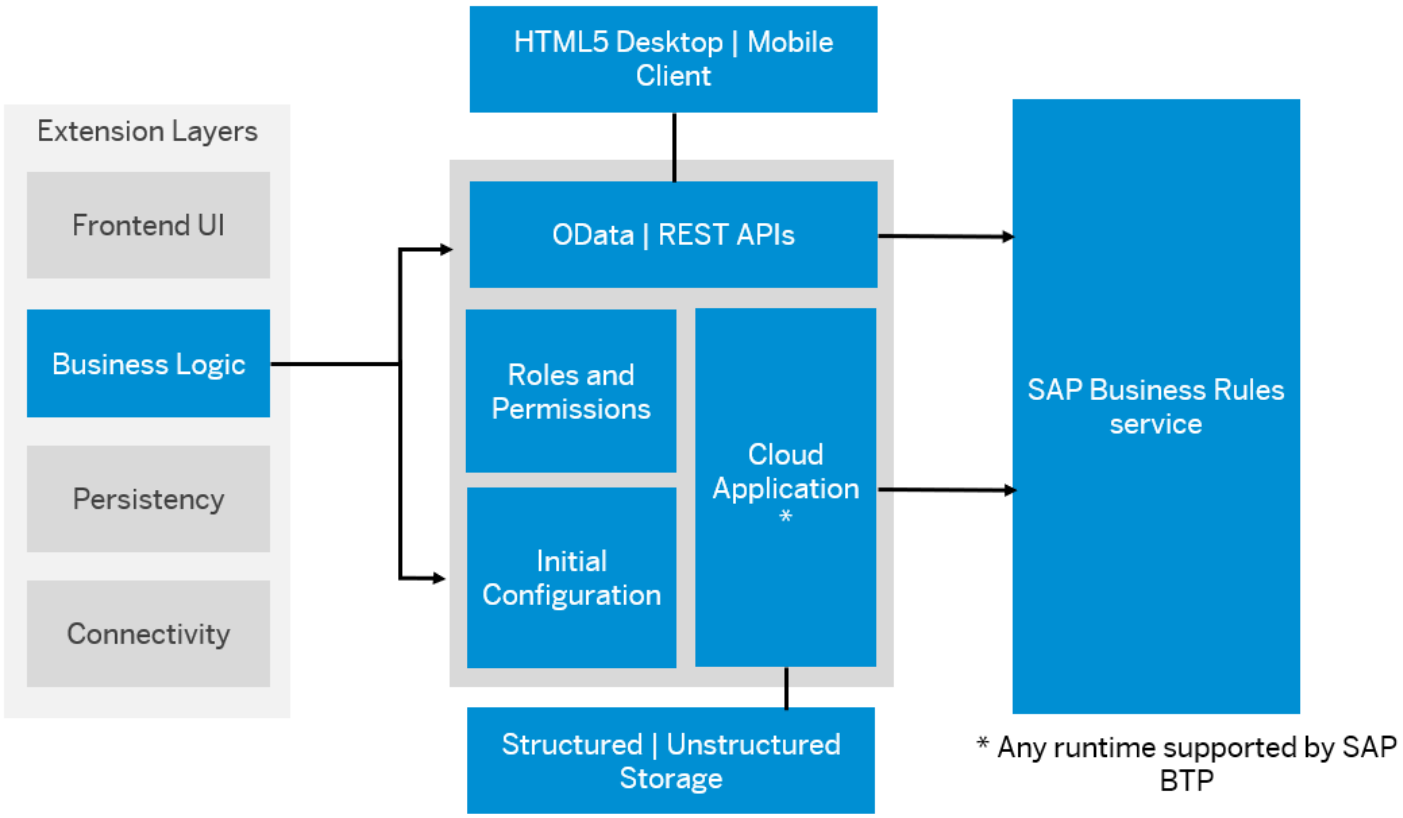
Decisions for Extension Workflows

EmployeeOnboardingScenario



An extension workflow enriches a core process running in a cloud application, like employee onboarding in SAP SuccessFactors, with customer-specific best practices. In most cases, a decision is required to enrich the context of the extension workflow. For example, company policy generally determines the details of a new employee onboarding workflow. It includes the assets like company car, mobile phone, and IT equipment to be assigned. This decision logic is represented by a business rule of decision table type, which is triggered from a workflow service task. For more information, see the **Configuring a Service Task** section in the SAP Workflow Service.

Back-End Business Logic for Fiori Extension Applications



Decoupled business logic makes it easier to develop, test, and operate extension applications on SAP BTP. It also enables the implementation of concepts such as zero-downtime updates or A/B testing for UI. It ensures that all security checks are performed, leaving no space for error when placing business logic in the UI level. Extension applications can consume any available SAP BTP runtime.

Example

The decision logic needed in the UI for discount computation based on promotional offers can be implemented using the business rules capability. You do not need to hardcode the frequent changes in business policies to the application layer,

which can impact the business logic for computing discounts. This reduces the effort for modifying the application logic based on business policy changes.

Line-of-Business Smart Process Applications

Smart process applications are a special category of application software. They are designed to support business activities that are people-centric, highly variable, loosely structured, and subject to frequent change.

A key ingredient of a smart process application is the flexibility in decision management. business rules capability plays an important role in flexible decision management by enabling line-of-business users to directly configure their business decisions. As an application developer, you can add this characteristic to your line of business application by embedding the business rules capability UI control in your smart process application. You can use it as business policy configuration like fraud detection policies, strategic procurement policies, or entitlement policies.

Expression Language 1.0

Expression language 1.0 is based on the intuitive rule expression language (REL), which enables users to define the conditions for triggering decision determination.

Readability is a key objective of REL as its aim is to enable business users to describe the complex conditions required to initiate decision determination.

The documentation includes examples taken from the vocabulary for a sample environment from the banking industry that is built on top of Business Rules Capability. The vocabulary describes the data model of the application in business language, similar to an entity relationship model in business applications.

Using the rule expression language, business users can describe the logical conditions that are evaluated. It can either be in real time when a triggering event (for example, an event from an Internet Web site) occurs, or offline in batch mode (for example, a scheduled run to analyze credit risk for accounts in a banking environment).

When formulating expressions using the rule expression language, the business user applies operators to operands that are actually data objects of the data model (application-dependent) as represented by the vocabulary. At runtime, the business conditions are evaluated, with the runtime context providing a set of initial values for the unambiguous object specification taken from the vocabulary.

Example

The following table lists the possible objects of a sample application for the banking environment:

Object Name	Description
`channel`	The medium from which a transaction is entered, such as teller, online banking, or mobile app
`customer`	The central object: bank customer
`customer_product`	A product that belongs to a specific customer, such as their account or loan.
`product`	A banking product, such as an account, loan, or mortgage.
`transaction`	An action applied by a customer to one of his or her products

This in-context object specification is what provides this language with its unique and simple expressive power.

Example

The following examples illustrate the context specification of the language:

- `annual_income of the customer`
  Explanation: The annual income of the current customer, where "annual income" is implemented as a field in the customer table of the vocabulary.
- `transaction_date of the transaction`
  Explanation: The date of the current transaction.

Quick Reference: Rule Expression Language

The following table shows the rule expression language at a glance.

Data Types Legend: B=Boolean D = Date N = Number S = String

Rule Expression Language

Operator	You May Also Use	Syntax (Examples)	Data Types
Logical Operators			
and		A = 10 and A< 5	B\D\N\S
or		A = 10 or A < 5	B\D\N\S
Comparison Operators (General)			
is equal to	=	A is equal to 10	B\D\N\S
is not equal to	!=	A is not equal to 10	B\D\N\S
is greater than	>	A is greater than 10	D\N\S
is less than	<	A is less than 10	D\N\S
is equal or greater than	>=	A >= 10	D\N\S
is equal or less than	=<	A =< 10	D\N\S
exists in		A exists in (5;10)	D, N, S
does not exist in		A does not exist in (5;10)	D, N, S
is between		A is between 0 and 100	D, N
is not between		A is not between 'd1' and 'd2'	D, N
Comparison Operators (String)			
is like		A is like 'John%'	S
is not like		A is not like 'John%'	S
contains		A contains 'John'	S
does not contain		A does not contain 'John'	S
starts with		A starts with 'John'	S
does not start with		A does not start with 'John'	S
ends with		A ends with 'son'	S
does not end with		A does not end with 'son	S

Data Types

Date type specifies a particular type for the values which is used in a rule expression language.

Using the rule expression language, you can create a condition by combining operands from the vocabulary with operators. The operands and operators are limited to certain data types, which are summarized in the following table:

Type	Cardinality	Values
Number	Scalar	Real numbers (with or without dot-decimal notation)
Boolean	Scalar	Boolean data-type: true, false
Date	Scalar	dd/mm/yyyy or the value 'today'
String	Scalar	Single-quoted, UTF-8 encoded
Timestamp	Scalar	Timestamps (date and time), UTC time

Conditions

A condition is a boolen expression. When creating rules, business users define one or more conditions in the service.

The condition describes the rule that determines whether the decision will be triggered.

Boolean Conditions

A Boolean condition is either a scalar Boolean selection, or an equality comparison with a Boolean constant.

Example

- `customer is employed`

  This condition can also be written as `customer is employed is equal to true`. However, this is redundant since the first part of the condition is already parsed as a Boolean statement.

- `customer is retired`
- `customer is not employed`
- `customer is not retired`

The following examples show Boolean conditions used with aliases:

- `age_calc is equal to false`
- `average_rating_check is equal to true`

Date and Timestamp

Date and Timestamp are the comparisons between the dates and/or time.

Date

Date constants are either predefined keywords (`today`, `tomorrow`, `yesterday`) or specified dates. Repository uses ISO 8601 format (YYYY-MM-DD) and the date format used in the user interface is based on the user locale.

If the expressions are created and updated using OData API, always use ISO 8601 format.

- Any `is between` comparison is equivalent to the respective `is after` and `is before` comparisons.

Example

`is between '12/06/2017' and '12/06/2017'` Explanation: Takes place later than or equal to 12/06/2017 00:00:00 and earlier than or equal to 12/06/2017 23:59:59.

Note

Use `is between` with a lower boundary value and an upper boundary value of the same data type. For example:

Following formats are supported for the input in the rule invocation:

1. YYYY-MM-DD
2. DD/MM/YYYY

Timestamp

Timestamp can be compared to date and there is no need to convert date into timestamp and vice versa.

Repository uses ISO 8601 format (YYYY-MM-DD HH:MM:SS.ssss) and the time stamp format used in the user interface based on the user locale. HH is in 24-hour format. If the expressions are created and updated using OData API, always use ISO 8601 format.

Example

- `is after '12/01/2017 23:59:59'`
  Explanation: Takes place after 12/01/2017 23:59:59.
- `is before '12/01/2017 00:00:00'`
  Explanation: Takes place before 12/01/2017 00:00:00.
- `is between '12/06/2017 00:00:00' and '12/06/2017 23:59:59'`
  Explanation: Takes place later than or equal to 12/06/2017 00:00:00 and earlier than or equal to 12/06/2017 23:59:59.

  #### Note

  Use `is between` with a lower boundary value and an upper boundary value of the same data type. For example:

  - Date is between Date A and Date B
  - Date is between Timestamp A and Timestamp B
  - Timestamp is between Date A and Date B
  - Timestamp is between Timestamp A and Timestamp B

The format supported for the input in the rule invocation is "YYYY-MM-DDTHH:MM:SS.ssssss".

Note

Dates and timestamps are stored and compared using UTC time zone only.

The rule expression language supports `is in the next`, `is in the last`, `is not in the next` and `is not in the last` operators.

`is in the last/next` N `minutes/days` includes the current minute/day respectively. So:

- `is in the last 2 days` is the same as `yesterday or today`
- `is in the next 2 days` is the same as `today or tomorrow`

Example

```
If
   LastPurchaseDate of the Customer is in the last 30 days
Then
   Discount = 10


If
   LastPurchaseTime of the Customer is in the last 1 hour
Then
   Discount = 20
```

The semantic meaning of these operators are

Expression	Semantics
flight_date of the flight `is in the next 3 days`	FLIGHT_DATE >= now and FLIGHT_DATE < now + 3 days
flight_date of the flight `is not in the next 5 days`	FLIGHT_DATE < now and FLIGHT_DATE >= now + 5 days
flight_date of the flight `is in the last 2 days`	FLIGHT_DATE <= now and FLIGHT_DATE > now - 2 days
flight_date of the flight `is not in the last 6 days`	FLIGHT_DATE > now and FLIGHT_DATE <= now- 6 days

Note

The `not` keyword negates the result.

Numeric Conditions

A numeric condition is a numeric comparison between two numeric expressions.

If one of the numeric expressions evaluates to null, the entire condition results in null, unless compared with the value null.

The numeric operators follow the usual (SQL-derived) operator precedence, which can be overwritten by using parentheses.

Example

- `credit_rating of the customer is not equal to 5`
- `credit_rating of the customer is between 5 and 8`

When calculating the count of an object, there is no need to specify the attribute of the object, as shown in the following examples. Note also how the `all` selection operator can vary in expressions and even change the entire meaning.

When using the `is between` and `is not between` operators on numeric values, the operands in the expression are included in the range of values evaluated.

Example

- `annual_income of the customer is between 10000 and 100000`
  Explanation: The annual income of customer is >=10000 and <=100000).
- `credit_rating of the customer is between 0 and 2`
  Explanation: The credit rating of customer is anywhere from 0 to 2 inclusive.

String Conditions

A string condition is the result of a logical string operation (substring, equality) on a string data object.

String comparisons are case-sensitive. If the string selection is non-scalar or null, then the result of the condition is null.

```
string-cond = ( string-operand compare-operator string-operand  ) | sub-string
```

Example

- `gender of the customer is equal to 'm'`
- `first_name of the customer starts with 'Joh'`
- `last_name of the customer does not contain 'Gold'`
- `first_name of the customer is like 'Joh%'`
  Where: "%" is an open wildcard, in other words, any name that starts with 'Joh'.
- `first_name of the customer is like '___n%'`
  Where: "_" is a one character wildcard, in other words, any name with 'n' as the fourth character.

Rule Expression Language in Decision Tables

In addition to being used in the text editor, the REL can also be used in decision tables. Decision tables are intended for "CASE-1...CASE-2...CASE-n" type rules, while text-based rules are used for "IF...THEN" type rules that have a single associated condition.

Using REL in a decision table is similar to using it in the context of textual expressions throughout this guide, with the following differences:

- The main attribute of a condition comes from the decision table's column header, which is an expression that is predetermined by the user who created the decision table. The operators and the rest of the conditions come from the cells content. There is no equals sign since there is a single value in the context of the column.

  ### Example

today – birthdate of the customer	gender of the customer	average_rating_check
18 years	'M'	true

  This is equivalent to
  `today - birthdate of the customer = 18 years and gender of the customer = 'M' and avera`

- Other conditions are also visualized differently within a decision table (for example, without the column header).

Example

credit_rating of the customer	annual_income of the customer
>8	is between 10000 and 100000
exists in (5;6;7)	>=100001

The first row in the table is equivalent to

```
credit_rating of the customer is greater than 8 and annual_income of the customer is be
```

The second row is equivalent to

```
credit_rating of the customer exists in (5;6;7) and annual_income of the customer >=100
```

- The rule can be modeled only with the expressions based on single leading objects.

Example

Age of the Employee	Location of the Company
=25	Bangalore

In this example, Column 1 is using `Employee` as leading data object and Column 2 is using `Company` as data object. However, with the enhanced parser-side validation checks, these exceptions are caught early in the rule validation instead during the rule service deployment.

With this modeling, the following error is highlighted:

Note

All expressions in the decision table need to have the same root data object.

If there is any association between the objects such as `Employee` and `Company`, namely: `Department`, and we must use `Employee` as the leading data object.

Then this example can be modeled as:

Column 1: Age of the Employee

Column 2: Location of the Department of the Employee

Examples

Users can enter complex conditions inside each table cell. For example:

credit_rating of the customer	annual_income of the customer
>=8 and first_name of the customer is not like 'John'	>12000 and gender of the customer = 'M'
>=5 and first_name of the customer is not like 'Jane'	<12000 and gender of the customer = 'F'

However, this is considered a misuse. A better approach is to create and fill the table as follows:

credit_rating of the customer	first_name of the customer	annual_income of the customer	gender of the customer

credit_rating of the customer	first_name of the customer	annual_income of the customer	gender of the customer
>=8	is not like 'John'	>12000	'M'
>=5	is not like 'Jane'	<12000	'F'

The rule expression language can also be used in the result columns of the decision table.

today - birthdate of the customer	Total amount	Output Param-1
>=18 years	is between 0 and 999	10
	is between 1000 and 2000	(credit_rating of the customer)*2

Here, the output column is `Output Param-1` and can be either a value or an expression. In this case, `10` or `(credit_rating of the customer)*2`, depending on the total amount in the second column.

Expression Language 2.0

You can model a rule using the Expression Language 2.0. This section gives you an overview about the features of this language.

Expression Language 2.0, which is based on DMN FEEL, provides a standard syntax for rule conditions, and reduces ambiguities while modeling a rule in business rules. Friendly Enough Expression Language (FEEL) is an element of the Decision Model and Notation (DMN) standard and is an improvised approach to modeling business decision logic.

Expression Language 2.0 can globalize the expression language by reducing translation efforts. This language can be used both in text rules as well as in decision tables.

ℹ Note

If your existing project is in rule expression language, you can change the expression language to Expression Language 2.0 using the migration feature. Choose **Migrate** from the **Details** tab of your project.

You can model the rule condition seamlessly by selecting the required vocabulary from the autosuggest list. For more information, see Autosuggest List in Expression Language 2.0.

Expression Language 2.0 supports the following business data types:

Business Data Type	Description
Number	Real numbers (with or without dot-decimal notation)
Boolean	Boolean data type: true, false
Date	The supported date format is ISO 8601 format (yyyy-mm-dd).
String	Single-quoted, UTF-8 encoded.
Timestamp	Timestamps (date and time), UTC time. The supported timestamp format is ISO 8601 format (yyyy-mm-ddThh:mm:ssZ).
Quantity	Codes for units of quantities. For more information, see Codes for Quantities.
Geometry	Geometrical expressions based on GeoJSON format.

Business Data Type	Description
Amount	Amount is expressed as the number of unit currency. The format of amount data type is <number> <currency code in ISO 4217 format>. **Example** 400 USD

Example

The following examples illustrate the syntax and context specification of Expression Language 2.0:

Typical operands in Expression Language 2.0 are as follows:

- **customer.annual income**:

  annual income is the attribute of the data object customer.

  Meaning: The annual income of the current customer.

- **customer.customer name**:

  customer name is the attribute of the data object customer.

  Meaning: The name of the customer.

A typical rule condition in Expression Language 2.0 with the above operands is as follows:

customer.annual income>=500000 AND customer.customer name MATCHES ‘John'

Meaning: The annual income of the customer is greater than or equal to 500000 and the customer name is ‘John'. Here, **customer** is the data object label, and **annual income** and **customer name** are the attribute labels of the data object **customer**. **AND** and **MATCHES** are the operators.

Related Information

Create a Project

Autosuggest List in Expression Language 2.0

Expression Language 2.0 lets you model a rule expression using the autosuggest list.

The vocabulary autosuggest list is a suggestion dropdown menu that lets you select the required element of the rule expression. You can also use free flow typing to enter the rule expression, which lets you type the rule expressions in the field and select the corresponding vocabulary elements from the autosuggest list.

The label of the entities are displayed in the autosuggest list. If the label is not maintained, then the name of the entitiy is displayed. The attribute of a data object is displayed in dot notation.

You can also view the business data type of the data object attributes like boolean(B), date(D), number(N), string(S), time(T), quantity(Q), amount(A), timestamp(U), and geometry(G), in the dialog for selecting attributes in the autosuggest list.

The features of the autosuggest list include:

- The suggestions are based on the business data type and only the relevant elements of the rule expression are displayed in the list.
- You can select the required operand, operator, or function. The operand can be a data object attribute or a rule.

  For more information, see

  - Operators
  - Functions
- You can provide the input values in the **Fixed Value** field.

  For more information, see Fixed Values.

  You can provide codes as the fixed value for certain quantities. For more information, see Codes for Quantities.

  Note

  - The autosuggest list is not displayed if the rule expression entered is incorrect.
  - In a decision table, the autosuggest list is not displayed if the rule expression ends with an operator.

Free Flow Typing

You can type the rule expressions in the field and select the corresponding vocabulary elements from the autosuggest list. Using free flow typing, you can also edit and delete the tokens in a rule expressions.

Note

You have to select the data object and attribute names from the autosuggest list. The expression is invalid if you type the data object and attribute names instead of selecting them from the autosuggest list.

You can also load the autosuggest list in between the data object or attribute names in a rule expression.

Example

In the following rule expression:

DO1.Equipment = 'Laptop' **AND** DO2.Date = 'Dec 3,2019'

- If the cursor is placed between **DO** and **1**, then all the data objects starting with DO are listed in the autosuggest list. Similarly, according to the cursor position, the corresponding data object or attribute name is listed.
- To change a date or timestamp value, edit the value in the **Fixed Value** field of the autosuggest list.

To delete any token in a rule expression, you can also use the BACKSPACE key or the delete key . If the token after the cursor position is a data object, attribute, a select or an aggregate function, then the token is deleted.

Example

In the following rule expression:

DO1.Equipment = 'Laptop' **AND** DO2.Price > 650

- If the cursor is placed before the attribute name **Equipment**, then the attribute name is deleted.

- If the cursor is placed in between or before the data object name, **DO1**, then both the data object and attribute name, **DO1.Equipment**, is deleted.

Related Information

Model a Rule Expression in Expression Language 2.0

Operators

You can use the following operators to model the rule expressions or conditions.

This section helps you understand the various operators used in expression language 2.0 and how to enter the operators using the correct syntax. The operators are listed in the autosuggest list only if they are relevant for the rule expression. You can use the operators on the following runtimes:

- Cloud
- ABAP on HANA
- HANA Classic

Business Data Types Legend: B=Boolean, D=Date, N=Number, S=String, T=Time, Q=Quantity, A=Amount, U=Timestamp ,G=Geometry

Mathematical Operators

Mathematical operators let you perform binary operations on data objects, fixed values, or rule expressions.

Operator Name	Operator Syntax	Example	Business Data Types
add	+	product.amount + product.taxvalue	N\S\Q
subtract	-	product.total - product.discount	N\Q
multiply	*	product.amount * product.taxrate	N\Q
divide	/	product.totalamount / COUNT(products)	N\Q

Logical Operators

Logical operators let you perform logical operations on data objects, fixed values, or between rule expressions. These operations return a boolean value.

Operator Name	Operator Syntax	Examples	Business Data Types
and	AND	customer.name = 'John' AND customer.country ='INDIA'	B\D\N\S
or	OR	customer.country = 'US' OR customer.country ='INDIA'	B\D\N\S
not	NOT	NOT('John')	N\S\D\T\B\Q

Comparison Operators

Comparison operators compare fixed values, data objects, or rule expressions and return **true** or **false**.

Note

For operands of business data type amount (A), you can only compare values of the same currency. Operations involving amount business data type can be performed only on cloud and ABAP runtimes.

Operator Name	Operator Syntax	Examples	Business Data Types
is equal to	=	order.productD = product.ID	N\S\D\T\B\Q\A
is not equal to	!=	order.productD != product.ID	N\S\D\T\B\Q\A
is greater than	>	wallet.balance > order.total	N\S\D\T\Q\A
is equal to or greater than	>=	wallet.balance >= order.total	N\S\D\T\Q\A
is less than	<	wallet.balance < order.total	N\S\D\T\Q\A
is equal to or less than	<=	wallet.balance <= order.total	N\S\D\T\Q\A

Range Operators

Range operators check for a particular value in a range of continuous values and return **true** if it is within the range. If not, they return **false**. You can choose to include or exclude the upper and lower values of the range.

Operator Name	Operator Syntax	Examples	Business Data Types
in	IN	customer.score IN (1..10)	N\S\D\T\B\Q
not in	NOTIN	customer.score NOTIN (1..10]	N\S\D\T\B\Q

You can use round brackets - '(' or ')' - to exclude the limit values and square brackets - '[' or ']' - to include the limit values.

Example

Example	Explanation
customer.score IN [1..10]	customer.score can be any number between 1 and 10, including 1 and 10.
customer.score IN [1..10)	customer.score can be any number between 1 and 10, excluding 10.
customer.score IN (1..10]	customer.score can be any number between 1 and 10, excluding 1.
customer.score IN (1..10)	customer.score can be any number between 1 and 10, excluding 1 and 10.

Array Operators

Array operators check for a particular value in an array of values and return **true** if the value is present in the array. If not, they return **false**.

Operator Name	Operator Syntax	Examples	Business Data Types

Operator Name	Operator Syntax	Examples	Business Data Types
exists in	EXISTSIN	customer.country EXISTSIN ['IND','GER','AUS']	N\S\D\T\B\Q
does not exist in	NOTEXISTSIN	customer.country NOTEXISTSIN ['IND','GER','AUS']	N\S\D\T\B\Q

For data objects with association, use the following syntax:

Cardinality	Operator	Operator Syntax	Examples
1..1	EXISTSIN	<Value/Attribute Name> EXISTSIN SELECT(<Values>)	Contact.Interaction **EXISTSIN**('WEBINAR', 'WEBSITE_VIDEO', 'WEBSITE_DOWNLOAD')
	NOTEXISTSIN	<Value/Attribute Name> NOTEXISTSIN SELECT(<Values>)	Contact.Interaction **NOTEXISTSIN**('WEBINAR', 'WEBSITE_VIDEO', 'WEBSITE_DOWNLOAD')
1..n	EXISTSIN	(<Value_1> EXISTSIN SELECT(<Table name>, <Column name>) OR <Value_2> EXISTSIN SELECT(<Table name>, <Column name>) OR .....<Value_n> EXISTSIN SELECT(<Table name>, <Column name>))	('WEBINAR' **EXISTSIN SELECT**(Contact.Interaction , Type) **OR** 'WEBSITE_VIDEO' **EXISTSIN SELECT**(Contact.Interaction , Type) **OR** 'WEBSITE_DOWNLOAD' **EXISTSIN SELECT**(Contact.Interaction , Type))
	NOTEXISTSIN	(<Value_1> EXISTSIN SELECT(<Table name>, <Column name>) AND <Value_2> EXISTSIN SELECT(<Table name>, <Column name>) AND.....<Value_n> EXISTSIN SELECT(<Table name>, <Column name>))	('WEBINAR' **EXISTSIN SELECT**(Contact.Interaction , Type) **AND** 'WEBSITE_VIDEO' **EXISTSIN SELECT**(Contact.Interaction , Type) **AND** 'WEBSITE_DOWNLOAD' **EXISTSIN SELECT**(Contact.Interaction , Type))

Functional Operators

Functional operators perform operations on strings or expressions that return a string value.

Operator Name	Operator Syntax	Examples	Business Data Types
matches	MATCHES	customer.name MATCHES 'Jo.*' customer.name MATCHES '*Dr\.*|*Mr\.*' **Note** Dot star (.*) represents any character.	S
does not match	NOTMATCHES	customer.name NOTMATCHES 'Jo.' **Note** Period (.) represents a character.	S
contains string	CONTAINS	customer.name CONTAINS 'Dr.'	S

Operator Name	Operator Syntax	Examples	Business Data Types
does not contain string	NOTCONTAINS	customer.name NOTCONTAINS 'Dr.'	S
starts with	STARTSWITH	customer.name STARTSWITH 'J'	S
does not start with	NOTSTARTSWITH	customer.name NOTSTARTSWITH 'J''	S
ends with	ENDSWITH	customer.name ENDSWITH 'J'	S
does not end with	NOTENDSWITH	customer.name NOTENDSWITH 'J'	S

Functions

You can use the following functions to model the rule expressions or conditions.

This section helps you to understand the functions in expression language 2.0 and how to use the functions in your rule expressions correctly. Functions are listed in the autosuggest list only if they are relevant for the rule expression.

Business Data Types Legend: B=Boolean, D=Date, N=Number, S=String, T=Time, Q=Quantity, A=Amount, U=Timestamp ,G=Geometry

ℹ Note

Functions involving amount (A) business data type can be used only on cloud and ABAP runtimes.

For more information on unified codes used for time related quantities such as seconds, minutes, hours, days, weeks or months, see Codes for Quantities.

Time and Duration Functions

Time and duration functions let you perform operations on a date, time, and timestamp values. You can select the date and timestamp values from the date and time picker in the **Fixed Value** section. You can also enter the date or timestamp in the **Fixed Value** field using the syntax given in this section.

Function Name	Function Syntax	Example	Business Data Types	Return Business Data Types	Supported Runtime
is in next	ISINNEXT(<test value>, <quantity value>)	ISINNEXT('2019-01-10','2 WK')	D\T\U	Boolean	Cloud, ABAP on HANA, HANA Classic
is not in next	ISNOTINNEXT(<test value>, <quantity value>)	ISNOTINNEXT('2019-01-10T22:00:00Z','2 HR')	D\T\U	Boolean	Cloud, ABAP on HANA, HANA Classic

Function Name	Function Syntax	Example	Business Data Types	Return Business Data Types	Supported Runtime
is in last	ISINLAST(<test value>, <quantity value>)	ISINLAST(process.start time, '20 MIN')	D\T\U	Boolean	Cloud, ABAP on HANA, HANA Classic
Today	TODAY()	Order.Shipdate = TODAY()	D	D	Cloud, ABAP on HANA, HANA Classic
Tomorrow	TOMORROW()	Order.Readydate = TOMORROW()	D	D	Cloud, ABAP on HANA, HANA Classic
Yesterday	YESTERDAY()	Order.Readydate = YESTERDAY()	D	D	Cloud, ABAP on HANA, HANA Classic
is not in last	ISNOTINLAST(<test value>, <quantity value>)	ISNOTINLAST(employee.birth year, '18 ANN')	D\T\U	Boolean	Cloud, ABAP on HANA, HANA Classic
add seconds	ADDSECONDS(<test value>, <quantity value>)	ADDSECONDS(Employee.LoginTime,15) ADDSECONDS('2019-01-10T22:00:00Z', 20)	D\T\U	D\T\U	Cloud
add minutes	ADDMINUTES(<test value>, <quantity value>)	ADDMINUTES(Order.Time, 10)	D\T\U	D\T\U	Cloud
add hours	ADDHOURS(<test value>, <quantity value>)	ADDHOURS(Employee.LoginTime, 2)	D\T\U	D\T\U	Cloud
add days	ADDDAYS(<test value>, <quantity value>)	ADDDAYS(License.EndDate, 30) ADDDAYS('2019-01-30', 30)	D\T\U	D\T\U	Cloud
add weeks	ADDWEEKS(<test value>, <quantity value>)	ADDWEEKS('2019-01-20', 14)	D\T\U	D\T\U	Cloud
add months	ADDMONTHS(<test value>, <quantity value>)	ADDMONTHS(License.ExpiryDate, 4) ADDMONTHS('2019-01-30',1)	D\T\U	D\T\U	Cloud
add quarters	ADDQUARTERS(<test value>, <quantity value>)	ADDQUARTERS('2019-01-30', 3) This function return a date three quarters after 2019-01-30, that is, 2019-10-30	D\T\U	D\T\U	Cloud

Function Name	Function Syntax	Example	Business Data Types	Return Business Data Types	Supported Runtime
add years	ADDYEARS(<test value>, <quantity value>)	ADDYEARS(Employee.YearOfBirth,2) This function returns a date, two years after the input date. ADDYEARS('2019-01-30T18:00:00Z', 4) This function returns a timestamp value of '2023-01-30T18:00:00Z'.	D\T\U	D\T\U	Cloud
subtract seconds	SUBTRACTSECONDS(<test value>, <quantity value>)	SUBTRACTSECONDS(Employee.LoginTime, 10) SUBTRACTSECONDS('2019-01-20T22:00:00Z', 90)	D\T\U	D\T\U	Cloud
subtract minutes	SUBTRACTMINUTES(<test value>, <quantity value>)	SUBTRACTMINUTES('2019-01-30T22:00:00Z', 10) SUBTRACTMINUTES(Order.ShipTime, 20)	D\T\U	D\T\U	Cloud
subtract hours	SUBTRACTHOURS(<test value>, <quantity value>)	SUBTRACTHOURS(Plant.ShutDownTime,4)	D\T\U	D\T\U	Cloud
subtract days	SUBTRACTDAYS(<test value>, <quantity value>)	SUBTRACTDAYS(Plant.OpenDate, 10)	D\T\U	D\T\U	Cloud
subtract weeks	SUBTRACTWEEKS(<test value>, <quantity value>)	SUBTRACTWEEKS('2020-10-10',7)	D\T\U	D\T\U	Cloud
subtract months	SUBTRACTMONTHS(<test value>, <quantity value>)	SUBTRACTMONTHS(Customer.SignUpDate,1)	D\T\U	D\T\U	Cloud
subtract quarters	SUBTRACTQUARTERS(<test value>, <quantity value>)	SUBTRACTQUARTERS('2020-10-10', 2) This function returns a date, 2020-04-10 which is 2 quarters before 2020-10-10.	D\T\U	D\T\U	Cloud
subtract years	SUBTRACTYEARS(<test value>, <quantity value>)	SUBTRACTYEARS(Vehicle.MaufacturingDate, 2) This function returns a date which is 2 years before the Vehicle.MaufacturingDate	D\T\U	D\T\U	Cloud
seconds between	SECONDSBETWEEN(<test value>, <quantity value>)	SECONDSBETWEEN(Employee.LogoutTime, Employee.LoginTime) SECONDSBETWEEN('2019-01-10T22:00:00Z','10-01-2019T12:40:00Z') SECONDSBETWEEN('2019-01-10','2019-01-08')	D\T\U	Number	Cloud
minutes between	MINUTESBETWEEN(<test value>, <quantity value>)	MINUTESBETWEEN(Employee.LogoutTime, Employee.LoginTime) MINUTESBETWEEN('2019-01-10T22:00:00Z','10-01-2019T12:40:00Z') MINUTESBETWEEN('2019-01-10','2019-01-08')	D\T\U	Number	Cloud
hours between	HOURSBETWEEN(<testValue>, <quantityValue>)	HOURSBETWEEN(Employee.LogoutTime, Employee.LoginTime) HOURSBETWEEN('2019-01-10T22:00:00Z','2019-01-09T12:40:00Z') HOURSBETWEEN('2019-01-10','2019-01-08')	D\T\U	Number	Cloud

Function Name	Function Syntax	Example	Business Data Types	Return Business Data Types	Supported Runtime
days between	DAYSBETWEEN(<test value>, <quantity value>)	DAYSBETWEEN(TODAY(), Emplyee.JoiningDate)	D\T\U	Number	Cloud
weeks between	WEEKSBETWEEN(<testValue>, <quantity value>)	WEEKSBETWEEN(TODAY(), Employee.JoiningDate)	D\T\U	Number	Cloud
months between	MONTHSBETWEEN(<test value>, <quantity value>)	MONTHSBETWEEN(TODAY(),Employee.DateofBirth) MONTHSBETWEEN('2019-01-10','2018-01-01')	D\T\U	Number	Cloud
quarters between	QUARTERSBETWEEN(<test value>, <quantity value>)	QUARTERSBETWEEN(TODAY(),Employee.DateofBirth	D\T\U	Number	Cloud
years between	YEARSBETWEEN(<test value>, <quantity value>)	YEARSBETWEEN(TODAY(),Employee.DateofBirth)	D\T\U	Number	Cloud
Second	SECOND(<test value>)	SECOND('2019-01-10T12:11:22.333Z') The function returns the seconds part of the timestamp, that is, 22.333	T/U	Number	Cloud
Minute	MINUTE(<test value>)	MINUTE('2019-01-10T12:11:22.333Z') The function returns the minutes part of the timestamp, that is, 11.	T/U	Number	Cloud
Hour	HOUR(<test value>)	HOUR('2019-01-10T12:11:22.333Z') The function returns the hour part of the timestamp, that is, 12.	T/U	Number	Cloud
Month	MONTH(<test value>)	MONTH('2019-01-10T22:11:22.333Z') The function returns the month part of the timestamp, that is, 01. MONTH('2019-10-15') The function returns the month part of the date, that is, 10.	D/T/U	Number	Cloud
Year	YEAR(<test value>)	YEAR('2019-01-10T22:11:22.333Z') The function returns the year part of the timestamp, that is, 2019. YEAR('2019-10-15') The function returns the year part of the date, that is, 2019.	D/T/U	Number	Cloud
Day of week	DAYOFWEEK(<test value>) Returns the day of the week with Monday having a value of 1. Value ranges between 1 and 7.	DAYOFWEEK('2019-01-10T22:11:22.333Z') The function returns a value of 4.	D/T/U	Number	Cloud
Day of month	DAYOFMONTH(<test value>) Returns the day of the month. Value ranges between 1 and 31.	DAYOFMONTH('2019-01-10T22:11:22.333Z') The function returns a value of 10.	D/T/U	Number	Cloud

Function Name	Function Syntax	Example	Business Data Types	Return Business Data Types	Supported Runtime
Day of year	DAYOFYEAR(<test value>) Returns the day of the year. Value ranges between 1 and 366	DAYOFYEAR('2019-01-10T22:11:22.333Z') The function returns a value of 10.	D/T/U	Number	Cloud

Select Functions

Select functions let you retrieve select values from a table. In the **Manage Rules Project** application, you can configure the select functions, using the autosuggest list, in the **Select Functions** configuration window. You can use select functions on the following runtimes:

- ABAP on HANA
- HANA Classic

Function Name	Function Syntax	Example	Business Data Types	Return Business Data Types	Supported Runtimes
top	TOP(table, number)	TOP (customer,5)	N\S\D\T\B\Q	Table	Cloud, ABAP On HANA, HANA Classic
select	SELECT(table, col1...coln)	SELECT(customer, Rating, Region)	N\S\D\T\B\Q	Structure/Table	Cloud, ABAP On HANA, HANA Classic
first	FIRST(table)	FIRST(customer)	N\S\D\T\B\Q	Structure	Cloud, ABAP on HANA

Aggregate Functions

Aggregate functions let you perform calculations on a set of values and return a single result value or a set of result values.

Function Name	Function Syntax	Example	Business Data Types	Return Business Data Types	Supported Runtimes
average	AVERAGE(table, column, indexcolumn1..indexcolumnN)	AVG (customer, orderAmount)	N\Q\A	Table\N\Q\A	Cloud, ABAP on HANA, HANA Classic
sum	SUM(table, column, indexcolumn1..indexcolumnN	SUM (customer, OrderAmount)	N\Q\A	Table\N\Q\A	Cloud, ABAP on HANA, HANA Classic
count	COUNT(table, indexcolumn1..indexcolumnN)	COUNT(customers)	N\S\D\T\B\Q\A	Table\N	Cloud, ABAP on HANA, HANA Classic
count distinct	COUNTDISTINCT(table, column)	COUNTDISTINCT(customers, region)	N\S\D\T\B\Q\A	N	Cloud, ABAP on HANA, HANA Classic
distinct	DISTINCT(table, column)	DISTINCT(customers)	N\S\D\T\B\Q\A	Table	Cloud, HANA Classic

Function Name	Function Syntax	Example	Business Data Types	Return Business Data Types	Supported Runtimes
minimum	MIN(table, column, indexcolumn1..indexcolumnN)	MIN(customer, orderAmount)	N\D\T\Q\A	Table\N\Q\A\D\T	Cloud, ABAP on HANA, HANA Classic
maximum	MAX(table, column, indexcolumn1..indexcolumnN)	MAX(customer, orderAmount)	N\D\T\Q\A	Table\N\Q\A\D\T	Cloud, ABAP on HANA, HANA Classic

Advanced Functions

Advanced functions can be used to perform advanced operations on string, mathematical, amount and geographical values.

Function Name	Function Syntax	Examples	Business Data Types	Return Business Data Types	Supported Runtimes
concatenate	CONCAT(<string1>,<string2>, <string3>.....<stringN>)	CONCAT('Firstname','LastName') CONCAT(Contact_S.ContactName , 'abc', 'def')	S	S	Cloud, ABAP on HANA, HANA Classic
round	ROUND(<number>,<value>)	ROUND(customer.credit_rating,2)	N	N	Cloud, ABAP on HANA, HANA Classic
power	POWER(<number>,<power>)	POWER(2,3)	N	N	Cloud, ABAP on HANA, HANA Classic
sin	SIN(<number>)	SIN(2)	N	N	Cloud, ABAP on HANA, HANA Classic
cos	COS(<number>)	COS(2)	N	N	Cloud, ABAP on HANA, HANA Classic
is within	ISWITHIN(<coordinates of the point>, <coordinates of the polygon>) **Note** Coordinates are in GeoJSON format.	ISWITHIN(customer.location , '{"type": "Polygon","coordinates": [[100.0, 0.0],[101.0, 0.0],[101.0, 1.0], [100.0, 1.0],[100.0, 0.0]]}')	G	Boolean	Cloud
is not within	ISNOTWITHIN(<coordinates of the point>, <coordinates of the polygon>) **Note** Coordinates are in GeoJSON format.	ISNOTWITHIN(customer.location , '{"type": "Polygon","coordinates": [[100.0, 0.0],[101.0, 0.0],[101.0, 1.0], [100.0, 1.0],[100.0, 0.0]]}')	G	Boolean	Cloud
is null	ISNULL(<Data object of type Element>)	ISNULL(Customer.ContactName)	All data types	Boolean	Cloud,ABAP On HANA, HANA Classic

Function Name	Function Syntax	Examples	Business Data Types	Return Business Data Types	Supported Runtimes
is not null	ISNOTNULL(<Data object of type Element>)	ISNOTNULL(Customer.ContactName)	All data types	Boolean	Cloud, ABAP On HANA, HANA Classic
is initial	ISINITIAL(<Data object of type Element>)	ISINITIAL(customer.name)	All data types	Boolean	ABAP On HANA
is not initial	ISNOTINITIAL(<Data object of type Element>)	ISNOTINITIAL(customer.name)	All data types	Boolean	ABAP On HANA
convert amount	CONVERTAMOUNT(<Amount>, <Currency code>)	CONVERTAMOUNT(customer.amount, 'EUR')	A	A (Element of type Amount)	Cloud, ABAP On HANA

Select and Aggregate Functions in Decision Tables in Spreadsheet

If you are modeling a decision table in Microsoft Excel, enter Select and Aggregate functions using the following syntax:

Function Name	Function Syntax	Example	Business Data Types	Return Business Data Types	Supported Runtimes
sort ascending	SORTASC(table,column)	Average(Filter(SortAsc(Customer, Name), Age > 30, player_height)	N\S\D\T\B\A	Structure/Table	ABAP on HANA, HANA Classic
sort descending	SORTDESC(table,column))		N\S\D\T\B\A	Structure/Table	ABAP on HANA, HANA Classic
filter	filter(table, condition)		N\S\D\T\B\Q\A	Structure/Table	ABAP on HANA, HANA Classic

Filter is the same as the Where field in aggregate and select function configuration windows.

Functions for Text Rule Execute Operations

The following execute operations and loop functions can be used while modeling a text rule without a result data object. You can use text rule execute operations and functions on cloud and ABAP runtimes.

For more information, see *Model a Text Rule for Execute Operations* in Model a Text Rule.

Execute Operations

Function	Syntax	Input	Description	Example
Update	UPDATE(<Target>, <Source>)	Target Entity	The target data object or attribute of type Structure or Element, that should be updated as per the value of the Source.	UPDATE(Customer.Employee Name, Employee.Employee Name)

Function	Syntax	Input	Description	Example
		`Source Entity` or `Source Expression`.	The value of the `Target Entity` is updated to the value of the `Source Entity` or the value returned by the `Source Expression`. The `Target` and `Source` should be of the same data object type. **Note** You can also use vocabulary rules as the source expression by selecting them from the autosuggest list.	The data object, `Customer.Employee Name` is updated as per the value of `Employee.Employee Name`.
Append	APPEND(<Target>, <Source>)	`Target Entity`	The target data object of type Table to which the `Source` has to be appended.	APPEND(`FlightTable, Flight`) The data object `Flight` is appended to the data object `FlightTable`.
		`Source Entity` or `Source Expression`	A data object that should be appended to the `Target` data object. The source data object should be of type Structure or Table or a rule that returns a data object of type Structure or Table.	

Loop Functions

Loop Function	Syntax	Input	Description	Example
For each	FOREACH(<Vocabulary>)	`Vocabulary` which is a data object of type Table or a rule that returns a data object of type Table.	You can also provide additional filter conditions for performing the execute operations	FOREACH(`FlightTable`)
		Single or multiple execute operations to be executed in a loop		

Fixed Values

You can provide the following types of data as the fixed value while modeling a rule expression.

Business Data Type of Fixed Value	Description	Examples of Rule Expressions

Business Data Type of Fixed Value	Description	Examples of Rule Expressions
Boolean	A Boolean condition is either a scalar Boolean selection, or an equality comparison with a Boolean constant.	**customer.average rating check = true** Explanation: Average rating check is set to true.
Numeric	A numeric condition is a numeric comparison between two numeric expressions.	**customer.credit rating != 5** Explanation: Credit rating of the customer is not equal to 5. **customer.credit rating IN [5...8]** Explanation: Credit rating of the customer is between 5 and 8.
String	A string condition is the result of a logical string operation (substring, equality) on a string data object. String comparisons are case sensitive. **Note** String values should be provided in single quotation marks in the **Fixed Value** field.	**customer.gender MATCHES 'm'** Explanation: Gender of the customer is 'm'. **customer.first name STARTSWITH 'Joh'** Explanation: First name of the customer starts with 'Joh'
Date	Date constants are either predefined keywords (today, tomorrow, yesterday) or specific dates. The date format used in the user interface is based on the user locale if it is selected from the autosuggest list. **Note** Date should be provided in ISO 8601 format(yyyy-mm-dd) in single quotation marks in the **Fixed Value** field.	**customer.login date > '2017-06-12'** Explanation: Customer login date is after '2017-06-12'. **customer.shipment date = TOMORROW()** Explanation: Customer shipment date is tomorrow's date.
Timestamp	Timestamps denote the date and time. Timestamps can be compared with dates and you do not need to convert a date into a timestamp or vice versa. **Note** Timestamps should be provided in ISO 8601 format(yyyy-mm-ddThh:mm:ssZ) in single quotation marks in the **Fixed Value** field. For more information on codes for units of time, see Codes for Quantities.	**ISINNEXT(customer.login time,'2 HR')** Explanation: If customer login time is '2019-01-10T22:00:00Z' (timestamp format), then the user checks if the login time is in the next two hours.
Quantities	Quantities can be provided in the form of codes. These codes denote units of quantities. **Note** For more information, see Codes for Quantities.	**ISNOTINNEXT(customer.login time,'10 MIN')** Explanation: The user checks if the customer login time is not in the next 10 minutes.
Geometry	Geometrical coordinates based on GeoJSON format. **Note** Geometrical coordinates should be provided in single quotation marks.	**ISWITHIN('{"type": "Point","coordinates": [125.6, 10.1] }' , '{"type": "Polygon","coordinates": [[[100.0, 0.0],[101.0, 0.0],[101.0, 1.0],[100.0, 1.0], [100.0, 0.0]]]}')** Explanation: The input location, {"type": "Point","coordinates": [125.6, 10.1] } is in the specified region.

Codes for Quantities

The following are the codes for units of measure:

Quantity	Unified Code	Example
Time		
days	D	ISINNEXT('2019-01-10','5 D') ISINNEXT('2019-02-12T01:54:45Z','2 D')
weeks	WK	ISINNEXT('2019-01-10','1 WK')
months	M	ISNOTINNEXT('2019-01-10','5 M') ISNOTINNEXT('2019-01-10T09:00:00Z','5 M')
years	ANN	ISINLAST('2019-01-10','2 ANN')
hours	HR	ISNOTINNEXT('2019-01-10T10:00:00Z','12 HR')
minutes	MIN	ISNOTINNEXT('2019-01-10T11:00:00Z','10 MIN')
seconds	S	ISNOTINNEXT('2019-02-12T01:54:45Z','5 S')
milliseconds	MS	ISINNEXT('2019-01-10','4 MS')
microseconds	US	ISINNEXT('2019-01-10','10 US')

Initial Setup

The tasks that you need to perform when setting up and enabling the business rules capability in a subaccount.

Prerequisites

- You are a global account administrator. See Getting a Global Account.
- You have created a space within a subaccount in which Cloud Foundry is enabled. See Managing Orgs and Spaces Using the Cockpit.
- Within the space, you are assigned to the `Space Developer` role for the subaccount. See Building Roles and Role Collections for Applications.

Context

You can set up the business rules capability using one of the following options:

- Automatic configuration using the Workflow Management booster
- Manual configuration and customization of the service

Using the Automatic Setup

The service can be set up automatically using the workflow management booster. The booster covers all services contained in the workflow management service and is available for both enterprise and trial accounts.

Procedure

1. Access your global account page in the SAP BTP cockpit, and choose **Boosters** from the navigation area.
2. On the **Set up account for Workflow Management** tile, choose **Start**.
3. Follow the wizard to choose the subaccount, organization, and space where the booster is executed.

   Running the booster successfully, does the following:

   - Sets up the launchpad for the following capabilities: Workflow, Business Rules, Process Visibility, and Workflow Management
   - Assigns entitlement and quota for Workflow, Business Rules, Process Visibility capabilities and Workflow Management service
   - Enables subscription to SAP Business Application Studio and Workflow Management
   - Creates the required service instance for the workflow capability, business rules capability, process visibility capability, and workflow management
   - Creates the following destinations: BUSINESSRULES_APIHUB, BUSINESS_RULES, WM_CF_SPACE_PROVIDER, SAP_API_Business_Hub, bpmprocessvisibility, bpmrulesruntime, bpmworkflowmanagement, bpmworkflowruntime, and bpmworkflowruntimeoauth
   - Assigns all role collections of workflow management including business rules capability

   Note

   Make sure that you configure user and password credentials for the `BUSINESS_RULES` and `BUSINESSRULES_APIHUB` destinations. The `BUSINESSRULES_APIHUB` destination is used when you import the rules from SAP API Business Hub, and the `BUSINESS_RULES` destination is used when you consume rules from other services. For more information on configuring destinations, see Create HTTP Destinations.

Related Information

Access the Manage Rule Projects Application

Using the Manual Setup

For use cases where you want to customize the setup of business rules capability, you can configure the service manually instead of running the booster.

Procedure

1. Access your global account page in the SAP BTP cockpit, and assign the entitlement for business rules capability with the respective service plan, to your subaccount.

Account Type	Service Plan
Enterprise account	standard
Trial account	lite

For more information, see Configure Entitlements and Quotas for Subaccounts.

2. Create a service instance of business rules capability. For more information, see Create a Service Instance of Business Rules Capability Using the Cockpit.

> **Note**
>
> You can also use the existing space for creating a service instance.

3. Optional: Create a service key for the service instance.

   For more information, see Create Service Keys Using the Cockpit.

4. Assign roles to your users.

   For more information, see Assign Roles to Your Users.

Related Information

Configure the Manage Rule Projects Application via HTML5 Application Repository
Trial Tutorial: Set up your account using Booster

Create a Service Instance of Business Rules Capability Using the Cockpit

Create a service instance to enable the business rules capability.

Context

You can create multiple service instances of business rules capability, across different spaces within the same organization. All service instances within the same organization share the same data.

> **Caution**
>
> If you delete the last remaining service instance of business rules capability across all the spaces in your organization, your data within the service is irrevocably erased.

Procedure

1. In SAP BTP cockpit, navigate to your subaccount.
2. In the navigation area, choose **Services** **Service Marketplace** .
3. In the **Business Rules** tile, choose **Actions** icon.
4. Choose **Create**.
5. In the **New Instance or Subscription** dialog, ensure that the correct service plan is selected.
   - If you are using a trial account, choose the **lite** service plan.
   - If you are using an enterprise account, choose the **standard** service plan.
6. Choose **Cloud Foundry** as the runtime environment.
7. Select a space in your Cloud Foundry org to create the instance.
8. Enter a name for your service instance, then choose **Create**.

   The new instance of the business rules capability is created in your Cloud Foundry space.
9. Optional: To view the service instance, either choose **Instances and Subscriptions** in the navigation area.
10. Optional: Create a service key for the service instance that you created.

> **Note**
>
> You need a service key to call the service APIs without a UI or for fetching the OAuth client credentials for integration with target runtime systems.
>
> For more information, see Create Service Keys in Cloud Foundry.

11. Optional: To bind a deployed application to the new instance of business rules capability, see Binding Service Instances to Cloud Foundry Applications.

> **Note**
>
> To consume the business rules capability in your application, you need to create a service instance of business rules capability, and bind the instance to your application.

Assign Roles to Your Users

Add roles to one or more role collections, and then assign these role collections to your users.

Prerequisites

- **User & Role Administrator** role is assigned to you, in the **Security** section of your subaccount.
- The users are stored in identity providers that are connected to SAP BTP.
  - Default identity provider (SAP ID service). For more information, see SAP ID Service.
  - Custom identity provider. For more information, see User Management in SAP Cloud Identity Service - Identity Authentication.

Context

You can define a role collection either by creating a new role collection or by copying an existing role collection. The copied role collection includes all the roles of the origin. However, it doesn't include the users or user groups. You can provide a new name and description to the copied role collection and assign users to it. You can find the copy in the list of role collections.

Define a Role Collection

Procedure

1. Navigate to your subaccount in the SAP BTP cockpit.
2. In the navigation area, choose **Security** **Role Collections**.
3. To create a new role collection:
   a. Choose + *Create New Role Collection*.
   b. Enter a name and optionally a description, then choose **Create**.
4. To copy an existing role collection:
   a. Choose **Copy** at the end of the row of an existing role collection that you want to use as a template.
   b. Enter a new name and optionally a description, then choose **Create**.

   You can find the new role collection created in the list of role collections.
5. Choose the new role collection, then choose **Edit**.
6. In the **Roles** tab, under **Role Name**search for the **Application Identifier** that begins with **bpmrulebroker**.

7. Choose the required role name and the corresponding role template from the dropdown list, and then choose **Add**. For more information on roles, see Authorization Configuration.
8. Optional: Keep adding roles or create additional role collections according to your requirements.
9. Choose **Save**.

> **Note**
>
> To delete a role collection, choose *Delete* at the end of the row of the role collection that you want to delete.

Assign Role Collections to Users

Procedure

1. Navigate to your subaccount in the SAP BTP cockpit.
2. In the navigation area, choose **Security** **Role Collections**.
3. Choose the role collection to which you want to assign users.
4. Go to the **Users** tab and choose **Edit**.
5. Enter the **ID** of the user to whom you want to assign the role collection to. If the user only exists in a connected identity provider, choose the identity provider and type in the e-mail address of the user.
6. If a user is not added to the respective identity provider, add the user by selecting **Add User** in the confirmation dialog box.
7. Optional: To add more users, choose + *Add a user*.
8. Choose **Save**.

   You have now assigned the user to the role collection. The user has all of the authorizations of the role collection.

Related Information

Authorization Configuration

Configure the Manage Rule Projects Application via HTML5 Application Repository

You should configure the **Manage Rule Projects** application using the HTML application repository to access the application.

Prerequisites

You have created a service instance of business rules capability and assigned the required roles.

For more information, see Initial Setup.

Context

Manage Rule Projects application lets you author, manage and deploy rules. Each business decision scenario can be automated and deployed to different systems, as rule projects, using the application. You should configure the **Manage Rule Projects** application by developing and deploying an approuter application using the HTML5 application repository and access it via SAP BTP cockpit.

Procedure

1. Develop and deploy an approuter application using the command line interface.

For more information, see Deploy an Approuter Application Using the Command Line Interface.

2. Access the **Manage Rule Projects** application via SAP BTP cockpit. For more information, see Access the Manage Rule Projects Application.

Deploy an Approuter Application Using the Command Line Interface

You should develop an approuter application using the MTA build tool and deploy it using the command line interface.

Prerequisites

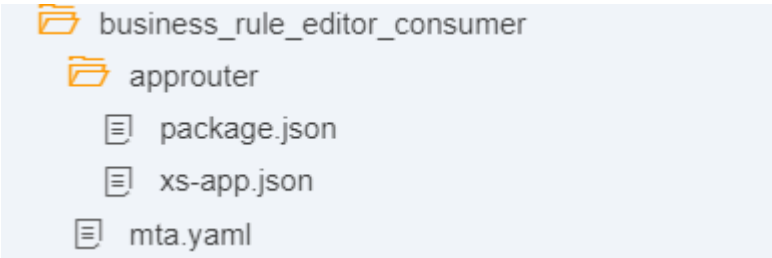
- You have an entitlement to application runtime.
- You have installed Java SE Runtime Environment 8 on your local machine.
- You have installed the Cloud Foundry command line interface from https://docs.cloudfoundry.org/cf-cli/install-go-cli.html .
- You have installed the multitarget application archive builder from https://tools.hana.ondemand.com/#cloud.
- You have installed the multitarget application Cloud Foundry CLI plugin from Multiapps CF CLI Plugin Documentation under **Download and Installation** section.
- You have installed the NPM command line interface tool from https://docs.npmjs.com/downloading-and-installing-node-js-and-npm .
- You have created a service instance of business rules capability. For more information, see Create a Service Instance of Business Rules Capability Using the Cockpit.
- You have created the custom domain and configured the application URL. For more information, see
  - Configuring Application URLs
  - Configuring Custom Domains

Context

You can create an approuter application using the MTA build tool and then generate an MTAR file using the Multi-Target Application Archive Builder. This MTAR file is then deployed in the Cloud Foundry environment using the command line interface. In SAP BTP Cockpit, bind the service instance of business rules capability to the approuter application.

Procedure

1. Create an `approuter` application folder with the files as given below.



2. Copy the following code to `package.json` file.

   package.json

```
{
"engines": {
"node": "^10.15.3"
```

```
},
"name": "ui-approuter",
"version": "1.0.0",
"dependencies": {
"@sap/approuter": "5.5.0",
"npm": "^6.9.0"
},
"scripts": {
"start": "node node_modules/@sap/approuter/approuter.js"
    }
}
```

3. Copy the following code to `xs-app.json` file.

   xs-app.json

```
{
    "welcomeFile": "comsapbpmrule.ruleeditor/index.html",
    "authenticationMethod": "route",
    "routes": [

    ]
}
```

4. Copy the following code to `mta.yaml` file.

   mta.yaml

   If you have not configured a custom domain, use the following `mta.yaml` file:

```
ID: bpm.rule.consumer.approuter
_schema-version: '2.0'
version: 0.0.1


parameters:
   deploy_mode: html5-repo
modules:
    - name: bpmruleconsumerapprouter
      type: javascript.nodejs
      path: approuter
      parameters:
        disk-quota: 256M
        memory: 256M
      requires:
       - name: bpmruleconsumer-html5appsrepo-apphost
       - name: bpmruleconsumer-uaa


resources:
 - name: bpmruleconsumer-html5appsrepo-apphost
   type: org.cloudfoundry.managed-service
   parameters:
      service: html5-apps-repo
      service-plan: app-runtime

 - name: bpmruleconsumer-uaa
```

```
    type: com.sap.xs.uaa
    parameters:
       config:
          xsappname: bpmruleconsumer-uaa-appname
          tenant-mode: dedicated
          role-templates:
             - name : UaaUser
               description: Role template for accessing SAP Business Rule APIs.Editor via user
               scope-references:
                 - uaa.user
```

If you have configured a custom domain, use the following `mta.yaml` file. Provide the `custom domain` details in the `modules` section, and the `application URL` that you have congured in the `resources` section.

```
ID: bpm.rule.consumer.approuter
_schema-version: '3.1'
modules:
- name: bpmruleconsumerapprouter
   parameters:
      disk-quota: 256M
      memory: 256M
      domain: <custom_domain>
      idle-domain: <custom_domain>
      routes:
        - route: <application URL>
   requires:
    - name: bpmruleconsumer-html5appsrepo-apphost
    - name: bpmruleconsumer-uaa
   type: javascript.nodejs
parameters:
   deploy_mode: html5-repo
resources:
- name: bpmruleconsumer-html5appsrepo-apphost
   parameters:
      service: html5-apps-repo
      service-plan: app-runtime
   type: org.cloudfoundry.managed-service
- name: bpmruleconsumer-uaa
   parameters:
      config:
         oauth2-configuration:
           redirect-uris:
             - https://bpmruleconsumerapprouter.<custom domain>/login/callback
         xsappname: bpmruleconsumer-uaa-appname
         tenant-mode: dedicated
         role-templates:
          - name: UaaUser
            description: Role template for accessing SAP Business Rule APIs.Editor via user t
            scope-references:
             - uaa.user
   type: com.sap.xs.uaa
version: 0.0.1
```

Note

Ensure that you have enough disk-quota and memory to deploy the approuter.

5. Use the command line interface to navigate to the `approuter` folder and execute the following commands.

```
npm config set @sap:registry https://registry.npmjs.org
npm install
```

For more information, see the SAP blog SAP NPM Registry.

6. Navigate to the parent folder (`business_rule_editor_consumer`) and execute the command.

```
java -jar <Path to mta jar downloaded from cloud tools> --build-target=CF build
```

For example, `java -jar C:\ABC\BPM\mta_archive_builder-1.1.7\mta_archive_builder.jar --build-target=CF build`

For more information, see Setting Up and Using the Multitarget Application Archive Builder.

7. Log on to the Cloud Foundry via command line interface and execute the following commands with the correct API endpoint.

```
cf api <API endpoint>
        cf login
```

Example

```
cf api https://api.cf.eu10.hana.ondemand.com
```

Note

From the **Overview** section in the SAP BTP cockpit, then navigate to the organization where you would like to deploy the MTAR file and copy the API endpoint.

8. Provide your credentials for authentication, then select the organization where the setup is to be done.
9. To deploy the MTAR file, execute the following command.

```
cf deploy <mtar-path>
```

Note

Ensure that there is only one uaa bound to your approuter application for deployment.

10. In the SAP BTP cockpit, navigate to **Service Instances**, then choose the service instance of type **business-rules**.
11. From **Actions**, choose , then choose the deployed approuter application from the **Applications** dropdown.
12. Choose **Save**.

Results

You can see the deployed approuter application in the **Applications** section of your space in the SAP BTP cockpit.

Related Information

Access the Manage Rule Projects Application

Access the Manage Rule Projects Application

After performing the required configuration, you can access the **Manage Rule Projects** application.

Prerequisites

- If you configured the service using the workflow management booster, make sure that the booster ran successfully. For more information, see *Using the Automatic Setup* in Initial Setup.
- If you have manually configured the service, the deployed approuter application is available in the **Applications** section of your Cloud Foundry space in SAP BTP Cockpit. For more information, see:
  - *Using the Manual Setup* in Initial Setup
  - Configure the Manage Rule Projects Application via HTML5 Application Repository

Context

You can create a project in the **Manage Rule Projects** application to implement the business logic using the entities. Each business decision scenario can be automated and deployed to different systems, as projects.

Procedure

1. If you have configured the service using workflow management booster, do the following:
   a. After the booster runs, choose **Go to Application** to access the workflow management launchpad.

   > **Note**
   > Alternatively, you can also access the launchpad from the Workflow Management application tile in the **Subscriptions** tab.

   b. In the workflow management launchpad, choose **Manage Rule Projects**.
2. If you have configured the service manually, do the following:
   a. In SAP BTP cockpit, navigate to **Applications**, then choose the application that you deployed.
   b. Choose **Restart**.
   c. To access the **Manage Rule Projects** application, select the URL from the **Application Routes** section.

Related Information

Create a Project

Configure a System

You should configure a system to consume business rules capability in runtime systems other than SAP BTP.

Prerequisites

- You have configured the **Manage Rule Projects** application via HTML application repository.

  For more information, see Configure the Manage Rule Projects Application via HTML5 Application Repository.

Context

A system represents a remote system where a rule service can be deployed. It contains information about target system in the form of destination and context path.

ℹ Note

It is not required to create a **System** to deploy a rule service locally into SAP BTP.

Procedure

1. In SAP BTP cockpit, configure the destination for the system. For more information, see Create HTTP Destinations.

   ### ℹ Note

   You can use the following authentication types to create destinations:

   - Basic Authentication
   - OAuth Client Credentials Authentication

   For configuring SAP API Business Hub as the managed system, provide the following information while configuring the destination:

   - **URL**: https://api.sap.com/.
   - **Authentication**: BasicAuthentication

   .

2. Navigate to the **Manage Rule Projects** application.
3. From the left pane, choose 👁 *Configure Systems*.
4. To create a new system, choose + *Add New System* , then perform the following substeps

   a. Provide the system **Name** and **Description**.

   b. In the **Destination** field, provide the name of the destination that you have configured in the SAP BTP cockpit.

   ### ℹ Note

   - For SAP HANA system, choose the destination which points to the URL of Rules HANA connector application.
   - For S/4 HANA system, choose the destination which points to the URL of the S/4HANA system.

   c. Choose the required context path from the **Context Path** dropdown list.

   - SAP HANA system
   - SAP S/4HANA system
   - SAP S/4HANA Cloud system

   ### ℹ Note

   You can also provide the context path manually and it should start with a slash. The context path is a relative path based on which destination is provided.

   - For SAP HANA system - **/rules-service/odata/rule_inbound_srv/**.
   - For SAP S/4HANA system - **/sap/opu/odata/sap/RULE_INBOUND_SRV**.

   /sap/opu/odata/sap/RULE_INBOUND_SRV is the default value for SAP S/4HANA Cloud system.

   d. Choose **Save**.

Development

Developer tasks for the business rules capability that are executed in the **Manage Rule Projects** application.

This image is interactive. Hover over each area for a description. Click highlighted areas for more information.

1. Access the **Manage Rule Projects** application.

   For more information, see Access the Manage Rule Projects Application.

2. Create a project.
3. Add data objects with attributes that represent your application context and optionally associate data objects based on mapping attributes.
4. Define the interface between your application and the rule runtime, and set the data objects to the rule service based on their usage.
5. Model your business logic using business rules. Define the condition constraints and the results to be returned for different business logic.
6. Configure the ruleset by grouping the related rules together and assigning them to a rule service.

Related Information

Concepts
Usage Scenarios
Create a Project
Deploy a Rule Service
Import a Project
Export a Project
Using Business Rules Capability APIs

Create a Project

Create a project in **Manage Rule Projects** application to implement the business logic using the business rule entities.

Context

A project is used to configure and manage the entities of business rules capability. A project can contain data objects, rules, rulesets, and rule services.

Features of a project include:

- **Label**: A label can be provided to the project and its entities along with the name and description for globalization. If a label is maintained for each entity of the project, the label name is displayed in the header of all the entities, breadcrumb navigation, and the autosuggest list. Labels can be translated into the user language, and lets the users model business rules in their language.
- **Version Management**: Create different versions and revisions of a project. Projects are categorized based on their working state as follows:

- **DRAFT**: The project that you are currently working on.
- **REVISED CONTENT**: The project that has been released and is ready for consumption.

  See Manage Project Versions and Revisions.

- **Project Hierarchy**: The project hierarchy feature lets you create a project with content that is separable. It includes the following functionalities:
  - Inclusion of other projects in your project to consume its exposed vocabulary.
  - Exposing your project vocabulary so that it can be consumed in other projects.
  - Extension of data objects of the included projects.

  An application provider can provide an application template as a project to the end user. End user can extend the template project to create an extension project with their own content. The extension project can then be included in the end user's local project to create a project that is independent of the template project content. This ensures content separation in each project.

If you've created a project in rule expression language, you can change the expression language to expression language 2.0 using the migration feature. For more information, see Migrate to Expression Language 2.0.

Procedure

1. In the **Manage Rule Projects** application, choose + *Add*.

   Note

   If you have created the project in any other system, you can import the project. See Import a Project.

2. In the **New Project** window, provide the name, label, and description of your project.

   Note

   Label is displayed as per the user locale. The default text of the **Label** is updated when you edit it in a particular locale.

3. From the **System** dropdown list, select the target system where you want to deploy your project.

   Note

   To deploy your project in any remote system, configure the system. See Configure a Managed System.

4. From the **Expression Language** dropdown list, choose one of the following options:

Option	Expression Language
1.0	**Rule expression language** See Expression Language 1.0.
2.0	**DMN SFEEL** See Expression Language 2.0.

   Note

   You cannot change the expression language once you have saved your project details.

5. If you use amount values in the ABAP runtime, you can enter the currency code in ISO 4217 format in the **Reference Currency** field.

   Note

   - Configure **Reference Currency** only if you want to perform operations on different currencies. The currencies are implicitly converted to the **Reference Currency** during the rule execution.
   - Ensure that the latest exchange rates should be available as part of TCURR table in the ABAP system.
   - **Reference Currency** is applicable only for projects in Expressions Language 2.0.

6. Optional: In the **Included Projects** field, include projects if you want to consume their exposed vocabulary in your project.

   See Include a Project.

7. To model the data objects of your project, see Data Objects .
8. To model the rule service for your project, see Rule Service.
9. To model the rules of your project, see Rule.
10. To define the rulesets of your project, see Ruleset.
11. Optional: Expose the data objects and rules of your project to other projects.

    See Expose Data Objects and Rules.

12. To check if your project is free of validation errors, choose **Validate**.
13. Choose **Save**.
14. To activate your project, choose **Edit**, then choose **Activate**.
15. Optional: To edit a project or an entity of the project, navigate to the respective tabs, then choose **Edit**.

    Note

    - Editing an entity creates a copy of that entity in inactive state, which can be modified. Once the inactive entity is activated, it replaces the previous active entity.
    - You cannot edit or delete a project or an entity, if it is locked by another user, until the session of the user expires. Each session expires in 20 minutes.

16. Optional: To delete a project or an entity of the project, navigate to the respective tabs, then choose **Delete**.

Related Information

Deploy a Rule Service
Manage Project Versions and Revisions
Export a Project

Data Objects

A data object is a reusable entity that represents the data domain of the consuming application.

A data object can be categorized based on its origin:

- **Local:** A data object holds the data of your project. Each data object can have many attributes. The data objects that are modeled in your project.
- **Included:** The data objects that are exposed by the included project.

Note

You cannot edit included data objects in your project.

Attributes add more information to a data object of type Structure. An attribute holds the data within a data object. You can also define a relation between data objects by creating an association between the source attribute and a target attribute. When an association is defined, the rule expression logic flows from the source attribute to the target attribute.

Types of Data Objects

Element

An element holds data of a particular business data type such as Number, String, Date, Boolean, Amount, Timestamp, etc.

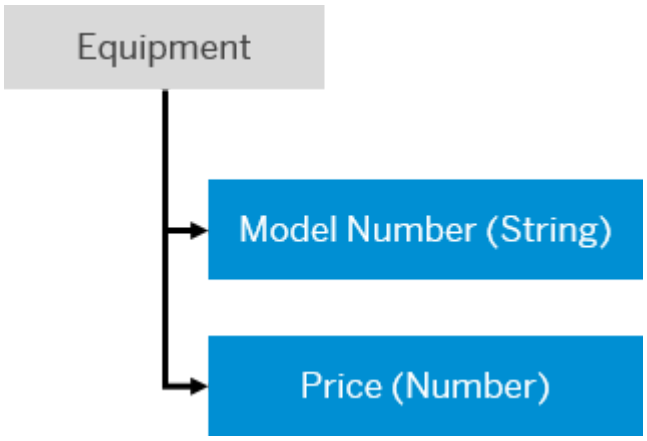
Example

Data object `Discount` of Element and business data type Number can have any numerical value such as 20, 30, 40, etc.

Structure

A structure consists of a set of attributes. Each attribute is similar to an element and consists of data of a particular business data type.

Example

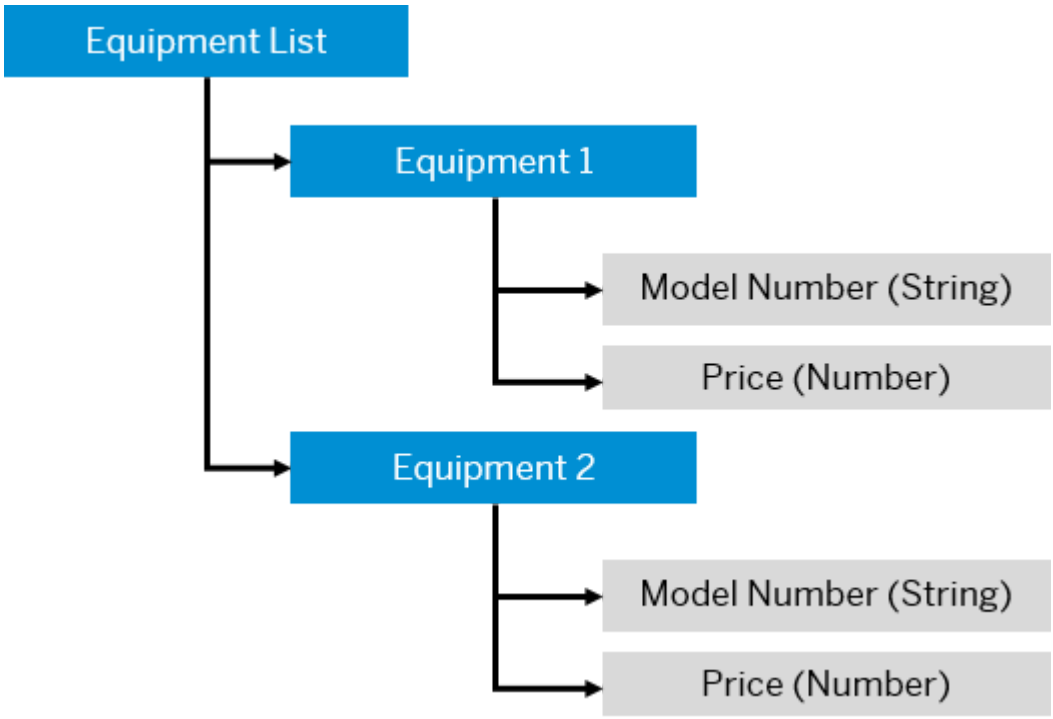


A data object `Equipment` can have multiple attributes that define its features like its `Model Number` and `Price`.

Table

A table has a set of structures. Each table type data object has a reference to a structure type data object called the reference data object. As soon as the reference data object is set, the attributes section of the table type data object gets populated with the attributes of the reference data object.

Example



A data object of type table `Equipment List` with reference to data object `Equipment` can store information about different types of equipment. Here, the reference data object will be `Equipment`. It also stores the specific information of each equipment via the reference data object.

Annotations

An annotation defines the context in which a data object would be used. It is the target runtime system of the data object. These annotations can be set at the level of a data object as well as an attribute.

- **No annotation**: SAP BTP is considered as a default annotation.
- **HANA DB**: If a data object is used in the context of SAP HANA system, then annotation type HANA DB should be assigned to it and its attributes. The following information needs to be added to the annotation at the data object and its attribute level.

Entity	Usage	Label	Description	Example
Data Object	Reference	Table/View Name	Name of the table/view.	Not applicable
Data Object	Reference	Schema	Schema where the table exists.	Not applicable
Data Object Attribute	Reference	Column Name	Name of the column.	Not applicable
Data Object Attribute	Input/Result	Data Type	HANA data type of the result column.	CHAR, VARCHAR
Data Object Attribute	Input/Result	Column Length	Length of the result column.	100
Data Object Attribute	Input/Result	Column Precision	Precision of the result column.	2
Data Object Parameter	Reference	View Parameter Name	The name of the input parameter of the CDS view. The value of View Parameter Name	If the input parameter is TEST, then the value should be $$TEST$$.

			should be provided within $$ and $$.	

- **ABAP CDS**

  If a data object is used in the context of S/4 HANA system (BRF plus), then annotation type **ABAP CDS** should be assigned to it and its attributes. The following information needs to be added to the annotation at the data object and its attribute level.

Entity	Usage	Label	Description	Example
Data Object	Reference	View Name	Name of CDS view of ABAP schema	Not applicable
Data Object Attribute	Reference	Column Name	Name of the column	Not applicable
Data Object Attribute	Input/ Result	Column Precision	Precision of the result column.	2
Data Object Attribute	Input/ Result	Column Length	Length of the result column.	100
Data Object Attribute	Input/ Result	Data Type	HANA Data type of the result column.	CHAR, VARCHAR
Data Object Parameter	Reference	View Parameter Name	The name of the input parameter of the CDS view of ABAP schema	Not applicable

- **HANA HDI**

  If a data object is used in the context of S/4 HANA system or SAP HANA System (into ABAP HDI Container or HDI Container directly), then annotation type **HANA HDI** should be assigned to the data object and its attributes. The following information needs to be added to the annotation at the data object and its attribute level.

Entity	Usage	Label	Description
Data Object	Reference	Container Name	The logical container name in which the HDI artifact is deployed
Data Object	Reference	View Name	Name of the view name (cube name)
Data Object	Reference	Entity Namespacex	Namespace in which the entity is defined
Data Object Attribute	Input/ Result	Column Precision	Precision of the result column. For e.g., 2
Data Object Attribute	Input/ Result	Column Length	Length of the result column. For e.g., 100
Data Object Attribute	Reference	Column Name	Name of the column
Data Object Attribute	Input/ Result	Data Type	HANA Data type of the result column. For e.g., CHAR, VARCHAR
Data Object Parameter	Reference	View Parameter Name	The name of the input parameter of the CDS view.

			The value of View Parameter Name should be provided within $$ and $$. For e.g., If the input parameter is TEST, then the value should be $$TEST$$.

Value Help

You can provide value help to attributes and data objects of type **Element** so that the values can be selected from the auto suggestion list while authoring a rule. The autosuggestion list provides the configured list of values as value help.

You can configure any of the following types of value help:

- **Value List** : The values are created and maintained in the business rules capability by creating the attribute or element value and description.
- **Service URL Mapping** : The values are maintained outside the business rules capability and you have to configure the managed system to consume the values from that system. For more information on configuring managed system, see Configure a System.

Note

- Press F4 to open the value help dialog while authoring a rule.
- To change the value help while authoring the rule in advanced mode, you can also click the value help attribute or element to open the value help dialog.

Example

In the following example, the ID of an equipment is an attribute. While authoring the rule, the value help is provided in the suggestion which has the value list with descriptions as shown in the table.

Value List

Code	Description
E001	Smartphone
E002	Laptop
E003	Tablet
E004	Smartphone

You can consume the attribute values in a rule by selecting it from the list.

```
If
ID of an Equipment is equal to 'E001'
Then
Order
```

- You can search for the value and description in the search bar.
- The advanced filter option is used to filter a value and description based on conditions which can be included or excluded. It can be applied for both value and description field.

For example:

If you want to filter the description `Smartphone` from the list, then the condition can be applied in both the value and description field as shown:

Include: Description is equal to `Smartphone`.

Exclude: Value is equal to `E004`.

The filtered result is `E001`, `Smartphone`.

- The value column can be sorted in both ascending and descending order.

> **Note**
> The search and filter options are case sensitive.

Related Information

Model Data Objects
Adding Attributes
Adding Associations
Adding Annotations

Model Data Objects

You can model the data objects that define the rule vocabulary. Data objects are used to model the business logic.

Context

A data object holds the data of your project. It is a reusable entity that represents the data domain of the consuming application. For more information, see Data Objects.

Procedure

1. From the **Manage Rule Projects** application, choose the required project.
2. Choose **Data Objects** **Local Data Objects**.

   > **Note**
   > The exposed data objects of the included projects are displayed in the **Included Data Objects** section.

3. To create a data object, choose + *Add*.

   > **Note**
   > A data object can be created or it can be imported from the managed system. To import the data object from the managed system, choose **Import** **Managed System**.
   >
   > For more information, see Import Data Objects from Managed System.

4. In the **Details** tab, provide the name, label, and description of the data object.
5. Optional: You can extend your data objects in other projects by adding attributes to it. To make your data object extensible in other projects, change the state of the **Extensible** toggle switch to **Yes**.
6. Select one of the following data object type from the **Type** dropdown list.
   - **Structure**

- **Table**
- **Element**

For more information, see Data Objects.

7. If you select the type **Structure**, then perform the following steps:
   a. Add attributes to your data object. For more information, see Adding Attributes.
   b. (Optional) Add associations between data objects. For more information, see Adding Associations.
8. If you select the type **Table**, then perform the following steps:
   a. Choose the **Reference Data Object** from the dropdown list.

> **Note**
>
> A table consists of set of data objects of structure type. This structure type data object is called the **Referred Data Object**. Attributes of the table type data object are the same as the attributes of the reference data object.

   As soon as you set the reference data object, the attributes section of the table type data object gets populated with the attributes of the reference data object.

   b. Optional: Choose the **Parameters** tab, then choose + *Add*.

> **Note**
>
> The parameter in the table defines the binding of a database to a form view. Whenever a table corresponds to a view with parameter in the database, then add a parameter of element type.

   c. In the new row, provide the **Parameter Name**, then choose the **Parameter Type**.
   d. Optional: Choose the annotation type for the parameter. For more information, see Adding Annotations.
9. If you select the type **Element**, then perform the following steps:
   a. Select a **Business Data Type** from the dropdown list
   b. Optional: To add value help for the element data object, see Adding Value Help.
10. From the **System for Value Help** dropdown, select the managed system from which you want to retrieve the value help for the data object.

> **Note**
>
> - For data objects of type Table, the system configured for the reference data object is the default system for value help.
> - If you want to configure a managed system for value help, see Configure a System.

11. Optional: Add annotation to your data object. For more information, see Adding Annotations.
12. To check if the data object is free of validation errors, choose **Validate**.
13. To activate your data object, choose **Activate**.
14. Choose **Save**.
15. Optional: You can extend an included data object by adding attributes or associations, using the **Data Objects** API. Extension is supported in the JSON payload of the API.

    For more information, see Rule Authoring API for Cloud Foundry.

> **Note**
>
> - You cannot further extend an extended data object.

- You can also override the annotation of the extended data object.

Adding Attributes

Attributes add more information to your data object. You can add attributes to your data object where each attribute corresponds to a data within the data domain of the data object.

Procedure

1. In the **Attributes** section, choose +.
2. In the new row, provide the name, label and description of the attribute.
3. From the **Business Data Type** dropdown list, choose one of the following business data type:
    - **Boolean**
    - **Date**
    - **Number**
    - **String**
    - **Timestamp**
    - **Geometry**

    Note

    Geometry business data type is supported only in expression language 2.0. For more information, see Expression Language 2.0.

4. Choose > in the new row.
5. Choose the **Value Help** tab, then add value help to the attributes. For more information, see Adding Value Help.
6. Choose the **Annotations** tab, then add an annotation type. For more information, see Adding Annotations.
7. Choose **Save**.

Adding Value Help

You can add value lists and service URL mapping as value help to an attribute or an element.

Context

While modeling a rule, you can select the required attribute or element value from the value help. The following are the types of value help:

- **Value List**: The help values are created and maintained in business rules capability.
- **Service URL Mapping**: The values are maintained outside business rules capability. You should configure the managed system and destination to consume these values from the system. The managed system returns a response JSON upon sending an API request to the service URL endpoint. Help values can be retrieved using any of the JSON keys.

Adding Value Lists

Procedure

1. From the **Value Help** section of your attribute or element, choose + .
2. From the dropdown list in the new row, select one of the following options :
    - **Value List**
    - **Service URL Mapping**

    > **Note**
    > You can create only one type of value help for an element or an attribute.

3. Choose > in the new row.
4. From the **Details** section, choose + .
5. Provide **Value** and **Description**.

    > **Note**
    > You can add multiple values in the value list.

6. Create a list of possible values of the attribute or the element, then choose **Save**.

    For more information about value lists, see Data Objects.

7. Choose **Save**.

Adding Service URL Mapping

Prerequisites

- You have to configure the destination and then the system. For more information, see:
    - Create HTTP Destinations
    - Configure a Managed System
- The service URL configured should provide the following support:
    - The service URL should be an HTTP endpoint based on either REST or OData Version 2.0.
    - When an API request is sent to the service URL, the response JSON should have the following keys:
        - **Value**
        - **Description** and **Language** keys if they are configured in business rules.
    - Service URL should support the following query options:
        - $filter for **Value**, **Description**, and **Language** keys.
        - $orderby for **Value** and **Description** keys.
        - $top and $skip

Procedure

1. From the **Value Help** section of your attribute or element, choose + .
2. From the dropdown list in the new row, select one of the following options :
    - **Value List**
    - **Service URL Mapping**

Note

You can create only one type of value help for an element or an attribute.

3. Choose > in the new row.
4. In the Details section, in the Relative Service URL field, provide the relative destination URL of the managed system.

Note

You need not provide the complete destination URL as the Relative Service URL.

Example

If the destination URL is `https://<host>/sap/c4c/odata/v1/c4codata/AccountABCClassification`, then `https://<host>/sap/c4c/odata/v1/c4codata` is the destination and `/AccountABCClassification` is the Relative Service URL.

5. Provide the JSON key value for which you want to retreive the help values in the Property Value field and then optionally provide a Property Description.
6. Optional: If you want to retreive help values in the language maintained in the managed system, then specify the language in the Property Language field.
7. Choose Save.

Adding Associations

You can add associations between your data objects.

Context

You can use associations to relate data objects to each other. You can add an association from a source attribute to a target attribute.

The following cardinality types are supported:

- **1..1 (oneToOne)**: For every record in a single object, there is one record in the associated object. This is the default cardinality type.
- **1..n (oneToMany)**: For every record in a single object, there are N matching records in the associated object.

Note

You can create associations only for data objects of business data types other than amount.

Procedure

1. Choose the Associations tab.
2. Choose +, then provide the name, label and description in the new row.
3. Select the Target Data Object Name from the dropdown list, then choose >.
4. In the Details tab, select the Cardinality.
5. In the Association Mappings tab, choose + , then select Source Attribute and Target Attribute.
6. Choose Save.

Adding Annotations

You can add annotations to your data objects to associate it to a target runtime system. Target runtime system is the context in which your data object or attribute would be consumed.

Procedure

1. Choose the **Annotations** tab, then choose +.
2. From the **Annotation Type** dropdown list, select the type of annotation.

   > **Note**
   >
   > - **No Annotation**: If the data objects is used in the context of cloud runtime.
   > - **HANA DB**: If the data object is used in the context of SAP HANA System.
   > - **ABAP CDS**: If the data object is used in the context of S/4 HANA system (BRF plus).
   > - **HANA HDI**: If a data object is used in the context of S/4 HANA system or SAP HANA System (into ABAP HDI Container or HDI Container directly).

3. If any annotation type is selected, then choose > and provide the information for assigned annotation type to the data object. For more information, see Annotations in Data Objects.

Import Data Objects from Managed System

You can import data objects from managed systems to consume them in your project.

Procedure

1. Navigate to the **Manage Rule Projects** application, then choose the project where you want to import the data objects.
2. Choose the **Data Objects** tab.
3. Choose **Import** **Managed system** .
4. From the **System** dropdown list, select the system from which you want to import the data objects.
5. Select a **Template**.
6. Select the data objects from the list or search for a data object.

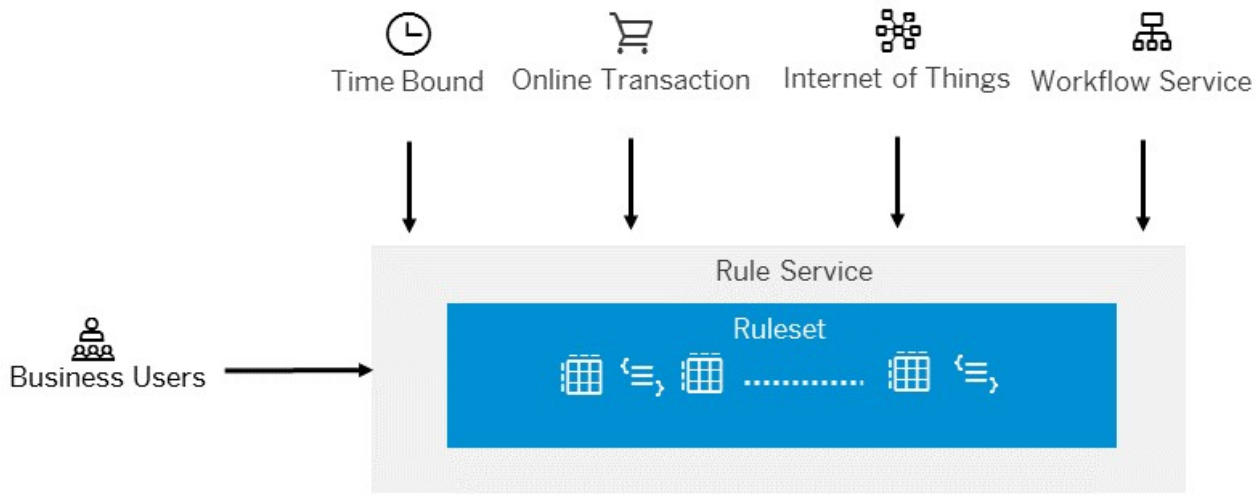
   > **Note**
   >
   > You can also append * with the search keyword to find the data objects that start with the same keyword.

7. Choose **Import**.

Rule Service

A rule service is an interface or an end point that enables an application to invoke a decision logic.

A rule service deploys a set of rules associated with it. You should invoke the rule service to implement the business decision logic in the consumer application. In an online transaction or workflow, rule service is invoked by passing relevant input from an application to the rule service.



Rule service is an interface that allows consumption of rulesets that an external application requires, by executing the business logic and returning the results to the application using a set of APIs.

A rule service enables an application developer to take a decision first approach using top down modeling. This starts by defining the vocabulary of the rule service. .

Rule Service Vocabulary

The vocabulary of the rule service, consists of data objects that are used as input and result. There are three types of rule service vocabulary:

Vocabulary	Description
Input	Data objects that are used as input for executing the rule service and provide the data needed to evaluate the rule constraints, by assigning annotations. The following are the types of annotations:
Result	Data objects that are used as results, provides a placeholder to return the outcome of the rule evaluation.
Reference	**Note** Reference is valid only for ABAP runtime Data objects that can be used as reference for the backend systems. See *Annotations* in Data Objects.

Example

Rule Service: Calculate Discount Rule Service	
Ruleset	Calculate Discount Ruleset

Rule Service: Calculate Discount Rule Service		

	Rule	Calculate Discount
		
		`If`
		`Equipment.Model Number = ‘A1V‘`
		`Then`
		`Discount = 30`

Input	Equipment	
Result	Discount	

For a rule service that is used to calculate discounts based on the details of an equipment, `Discount` can be configured as the result and `Equipment` can be configured as the input. When the rule service is deployed onto a system and invoked, the rule `Calculate Discount` linked to the ruleset `Calculate Discount Ruleset` of the rule service is executed. If the condition is satisfied, the data object `Discount` with the discount value 30 is returned.

Annotations for Rule Service

A rule service can be deployed on to different runtimes after modeling in SAP BTP

- **No annotation**: SAP BTP, by assigning is considered as the default annotation.
- **ABAP on HANA** : To deploy a rule service on S/4HANA System (BRF plus).
- **HANA XSC**: To deploy a rule service on SAP HANA system.

Platform Type	Label	Description
Annotation type	Package	Package on the target system where .hdbprocedure and its related artifacts are generated.
	Schema	Target schema in which to generate the runtime SQL procedure.

- **HANA Advanced**: To deploy a rule service into ABAP HDI container of S/4HANA system or HDI container of SAP HANA system.

Annotation type	Label	Description
HANA Advanced	Source Namespace	Namespace in which to generate HDI artifacts.
	Source Container	Name of the HDI container in which rules HDI artifacts will be deployed.
	Target Namespace	Namespace in which the synonym config for the rules SQL procedure needs to be generated.
	Target Container	HDI Container in which the synonym config for the rules procedure needs to be generated.
	Target Synonym	Name of the synonym for which the synonym config needs to be created

		pointing to rules SQL procedure.

Related Information

Model a Rule Service

Model a Rule Service

You can define the rule service to deploy a set of rules.

Prerequisites

You have created the data objects to be included in the rule service. For more information, see Model Data Objects.

Context

You can deploy a set of rules through the rule service. Rule service is an interface between the **Manage Rule Projects** application and the consumer application.

Procedure

1. Navigate to the **Rule Services** tab of your project, then choose ＋ .
2. In the **New Rule Service** window, provide name, label and description for the rule service.
3. Choose the **Vocabulary** tab, then perform the following steps:
    a. Choose ＋.
    b. Choose the **Name** from the dropdown list.
    c. From the **Usage** dropdown list, choose one of the following type:
        - **Input**
        - **Result**
        - **Reference**

    ### ⓘ Note

    - All data object types can be used as **Input**.
    - If the rule service returns more than one result, then the result data object should be of **Table** type.
    - All the data objects and associated data objects used in the expression should be passed explicitly as **Input** while modeling the rule service.
    - You have to create an **Element** for each table parameter and add it as **Input** for the rule service.

4. Add annotation to your rule service. For more information, see Adding Annotations.
5. Optional: If your rule service result is of business data type timestamp in cloud runtime, then you can configure the timestamp format to be displayed. To configure the timestamp format, do the following:
    a. Choose the annotation that you have configured.
    b. In the **Details** section, from the **Timestamp Format** dropdown, choose one of the following options:
        - **UTC Date Time**: Timestamp in `yyyy-mm-ddThh:mm:ssZ` format.

- Local Date Time: Timestamp in `yyyy-mm-ddThh:mm:ss` format.

c. Choose Save.

> **Note**
>
> Even when the annotation is not configured for the rule service, Local Date Time is the default format.

6. Choose Validate to check for the validation errors.
7. To activate your rule service, choose Activate.
8. Choose Save.

Rule

A rule is the technical representation of a simple business logic to be applied to a business case.

A rule defines a business logic that, once evaluated against live data, leads to a decision. A rule includes one or more conditions, and outputs that are triggered upon satisfaction of the condition or conditions. Rules can be adapted to the increasing complexity of business cases by combining any number of rules to a rule set.

> **Note**
>
> Rules can be triggered only in the context of a rule set to which a rule is assigned. They cannot be triggered as standalone objects.

Decision Table

A decision table is a collection of input and output rule expressions in a tabular representation. Multiple rule conditions can be configured using a decision table. The following is an example of a decision table rule:

> **Example**

[Conditions]		[Result]	
Designation of Employees	**Position of Employee**	**Device**	**Mobile Device**
Manager	T4	A1 Laptop	A1 Mobile
Manager	T3	A2 Laptop	A2 Mobile
Developer	T2	A3 Laptop	NA

During onboarding an employee's equipment is allotted based on his or her employee designation and position in the organization. In the above table, the designation and position of an employee is shown in the **[Conditions]** column. When the conditions match, the result that is specific to that match is executed.

Text Rule

A text rule is the collections of rule expressions in simple if-then format. It has the following structure:

```
IF
   Condition
THEN
   Operations
ELSE IF
```

```
    Condition
 THEN
    Operations
 ELSE
    Operations
```

When `condition` evaluates true, `then` operations get executed otherwise `else if` or `else` operations get executed (if provided). `Else` and `Else If` operations are optional. Multiple `Else If` statements can be added in a condition using **Add Else If** option.

Conditions

The condition should meet one of the following:

- A data object of boolean type
- An expression with a result of boolean type
- A simple range expression

Operations

The following is an example for a rule to produce a result

```
Rule result: Structure1 with components C1 and C2
Rule body:
IF age > 18
THEN Structure1-C1 = OK, Structure1-C2 = 1
```

The following is an example for a rule to work in context

```
Rule result: none (assuming Structure 1 is in execution context|business vocabulary)
Rule body:
IF age > 18
THEN Update Structure1-C1 = OK, Update Structure1-C2 = 1
```

Basic and Advanced Mode

ℹ Note

Modes are applicable only for Rule Expression Language 1.0.

- The **Basic** mode is explained as follows
    - If the operator is not set in the settings of the decision table, then the operator list will be provided as a dropdown in the popover of the decision table row. Once the operator is selected, you will get the input field option to set the value for the column.
    - Different options are provided to set the values based on the selected data type such as, input field for **string** and **number**, date picker for **date**, dropdown for **boolean**, time picker for **time**.
    - If the expression is given in the input field for the data type string and number, then the expression can be validated by clicking on the popover itself. If the data is wrong, then the expression will not be taken and the popover gets closed.

- If the data type is string, the value can be entered without quotes.
- **Exist in** operator is not supported.

- The **Advanced** mode enables you to create a rule with the auto suggestion and value help feature, which is applicable only for result column.

Vocabulary Rules

Note

Vocabulary rules are supported only in expression language 2.0.

Vocabulary rules are reusable rules which can be used in other rules. You can include both text and decision table rule as a vocabulary rule. The result returned by the vocabulary rule can be consumed in a rule expression.

To use vocabulary rules while modeling a rule expression, you should add the vocabulary rules and the relevant data objects as a part of its ruleset vocabulary. When you add a vocabulary rule to the ruleset vocabulary, the rule is listed in the autosuggest list for modeling a rule expression. For more information, see Define Rulesets.

Related Information

Model Rules

Model Rules

You can model rules to create the business logic required to deploy a rule service.

Prerequisites

You have created the rule service for your project. For more information, see Model a Rule Service.

Context

A rule is the technical representation of simple business logic that, once evaluated against live data, leads to a decision. One or more rules can be used to represent an end-to-end business scenario. It can be modeled in business rules and deployed to your application via a rule service. You can either create a new rule or copy a rule and make changes to it. For more information, see Rule.

Procedure

1. Navigate to the **Rules** tab.
2. To create a new rule, do the following:
   a. In the **Local Rules** section, choose +.
   b. In the **New Rule** window, provide a name for the rule.
   c. To model a decision table, choose **Decision Table** from the **Type** dropdown list, and then choose **Create**. For more information, see Model a Rule in Decision Table.

   ### Note

   If you are modeling a rule in expression language 1.0, then choose one of the following modes from the **Mode** dropdown list:

   - **Advanced**

- **Basic**

For more information, see Rule

d. To model a text rule, select **Text Rule** from the **Type** dropdown list, then choose **Create**. For more information, see Model a Text Rule.

3. To create a copy of a rule, see Copy a Rule.

Copy a Rule

Copy an existing rule instead of creating a new rule.

Context

Instead of creating a new rule, you can create a copy of an existing rule and add your changes. You can use copied rules in scenarios that require multiple rules with minor changes in business logic.

Procedure

1. In the **Rules** tab, choose *Copy Rule*.
2. In the **Copy Rule** window, provide a **Name** and **Description** for the new rule.
3. From the **Copy From** dropdown list, choose the rule from which you want to create a copy.
4. Choose **OK**.

   You are redirected to the new rule overview page.
5. After editing the new rule, choose **Save** or **Activate**.

Model a Rule in Decision Table

You can model the rule conditions in tabular format as a decision table.

Prerequisites

You have selected the rule type as **Decision Table**. See, Model Rules.

Context

A decision table consists of the following:

- **Condition**: One or more conditions to be met; the conditions are defined within a decision table.
- **Result**: Output values that are returned upon successful evaluation of the condition or conditions.

You can specify how the rule engine should fetch the result. The two types of hit policies are the following:

- **First match**: The rule engine fetches the first occurrence that matches the condition and returns it as result.
- **All match**: The rule engine fetches all the occurrences that matches the condition and returns them as result.

A decision table can be exported to your local system as a spreadsheet and rule expressions can be modeled outside the **Manage Rule Projects** application. This spreadsheet can then be imported into the **Manage Rule Projects** application.

This functionality lets the business users work independent of the application and use spreadsheet, which they are more comfortable with, to modify the business logic. This file can then be shared with an application developer to import it into the application.

Procedure

1. Provide the label and description for the rule.
2. Choose the **Decision Table** tab.
3. To configure conditions, choose ⚙.
4. From the **Hit Policy** dropdown list, choose one of the following:
   - **All Match**
   - **First Match**

   Note

   If the **Hit Policy** is **All Match**, the result data object should be of **Table** type.

5. In the **Condition Expression** field, provide the rule expression.

   Note

   - If you have selected the **Expression Language** as 1.0, provide the rule expression in rule expression language 1.0. For more information, see Expression Language 1.0
   - If you have selected the **Expression Language** as 2.0, model the rule expression in expression language 2.0 (DMN SFEEL). For more information, see Model a Rule Expression in Expression Language 2.0.

6. (Optional) In the **Label** field, provide a name of your choice as an identifier for the condition expression.

   Note

   We recommend that you to provide a label for each condition as the complete condition expression is not displayed in the decision table column header. To view the complete condition expression, hover the mouse pointer over the label.

7. Choose the required operator from **Fixed Operator**.
8. To add multiple conditions, choose +.
9. From the **Result** dropdown list, choose the result data object.
10. Optional: If you want to update the attributes of the selected result data object before applying the settings, choose the ↻.

    Note

    The refresh data object feature reads the attributes of the selected data object and automatically fetches the predefined result attributes.

11. Choose one of the following access modes for the result attribute:
    - **Hidden**
    - **Editable**

    Note

    - The **Hidden** access sets that default value to all the rows corresponding to the attribute in the decision table and the result column gets hidden. The default value is mandatory.

- The **Editable** access sets that default value to the new rows, which is created after the settings are applied. The default value is optional. When the result data object changes, by default all the attributes has the access mode as editable with no default value.

12. Choose **Apply**.
13. Provide the fixed values, rule conditions or expressions in the columns of the decision table row. To add more conditions, choose **Add Row** **Insert First** .

Note

- To add a row below a row, select the row and then choose **Add Row** **Insert After** . By default, the row is added to the top of the table.
- If you do not have input values for a given cell in a row, leave the cell empty.

Example

If you do not have any value for **Equipment Type** in the first row, you can leave the cell empty as shown:

If				Then		
countryofCompany	**company of the Employee**	**jobTitle of the Employee**	**Employee is isFulltimeEmployee**	**Currency**	**Equipment Type**	**Equipment Price**
is equal to 'USA'	is like is 'SAP'	is like is '%'	is equal to true	'EUR'		35.96
is equal to 'DEU'	is like is 'SAP'	is like is '%'	is equal to true	'EUR'	'Notebook'	76.98

14. To delete a condition from the decision table, select the row, then choose **Delete Row**.
15. To edit rows in the decision table, select the required row, then choose the required option from the following:
    - **Copy Row**
    - **Cut Row**
    - **Paste Row**
16. Optional: Choose **Validate** to check whether the rule modeled is properly validated or not, else check the validation errors.
17. To activate the decision table rule, choose **Activate**.
18. Choose **Save**.

Note

- To export a decision table rule to your local system, see Export a Decision Table.
- To import a decision table rule from your local system, see Import a Decision Table.

Export a Decision Table

You can export an existing decision table in the **Manage Rule Projects** application as a spreadsheet to your local system.

Prerequisites

You have created and validated the rule to be exported. For more information, see Model a Rule in Decision Table.

Context

If the business user wants to modify the current implementation of business logic, an application developer can provide the existing decision table rule to the user by importing it to the local system.

> **Note**
> Only the decision table columns set as **Editable** will be exported.

Procedure

1. Choose the decision table rule you want to export.
2. Choose **Export**.
3. Save the decision table rule as a spreadsheet in your local system.

Import a Decision Table

You can import a decision table as a spreadsheet to an existing decision table in the **Manage Rule Projects** application.

Prerequisites

- You have created the decision table in the **Manage Rule Projects** application.

  For more information, see Model a Rule in Decision Table.
- The spreadsheet has the Rule ID and Project ID.
- The headers in the decision table in the editor matches with the headers in the spreadsheet.
- You have added an apostrophe before a string and a date.
- You have provided the values for all the cells in the spreadsheet. For any cell that is empty, provide hyphen as the value.
- If your project is in expression language 2.0, ensure that you have entered the operators and functions in the correct syntax. The import will be unsuccessful if the syntax is incorrect. For more informaton on the syntax, see Operators and Functions in Expression Language 2.0.

Context

Business users can define the business logic in a spreadsheet, and an application developer can import this file into the **Manage Rule Projects** application. This gives flexibility to the business users, who may not be aware of the application functionalities, to define the logic that is independent of the application.

Procedure

1. Navigate to **Manage Rule Projects** application.
2. Choose **Rules** **Local Rules**.
3. Choose the decision table you want to import, then choose **Import**.
4. Browse your local system to select the spreadsheet containing the decision table rule.

Model a Text Rule

You can model a rule in a simple if-then format. You can also model a text rule to perform operations on data objects or attributes in expression language 2.0.

Prerequisites

You have selected the rule type as **Text Rule**. See, Model Rules.

Context

A text rule is the collection of rule expressions in a simple if-then format. For more information, see *Text Rule* in Rule. You can model a text rule in two different ways:

- **Text rule with a result data object**:

  You can model a text rule with or without a result data object. A text rule that stores the return value in a result data object is supported in expression language 1.0 and 2.0.

- **Text rule for execute operations**:

  When you do not assign a result data object to a text rule, you can perform execute operations on existing data objects or their attributes. Execute operations are supported only in expression language 2.0. You can update or append values to the data object. You can also add multiple execute operations in a single text rule. Text rule execute operations can be used with loop functions like **FOREACH** to perform execute operations in a loop.

  Text rule execute operations can be used together with the orchestration ruleset execution policy to implement execute operations in a specific order. To do this, you select the **Orchestration** execution policy while modeling the rulesets. For more information, see Ruleset.

Model a Text Rule with a Result Data Object

Procedure

1. Provide a **Name**, **Label**, and **Description** for the rule.
2. In the **Text Rule** tab choose ⚙.
3. From the **Result** dropdown list, choose the result data object.
4. Optional: Choose ↻ if you want to update the text rule results of the selected result data object before applying the settings. This reads the attributes of the selected data object and automatically fetches the predefined result attributes.
5. Choose one of the following access modes for the result attribute:
   - Hidden: Sets the default value to the attribute of the data object in the text rule, where it is then hidden from the text rule.
   - Editable: Sets the default value to the new entries of data object that is created after the settings are applied and a new **Else** block is created. When the result data object changes, all of the attributes have the **Editable** access mode with no default value.
6. Choose **Apply**.
7. Provide the rule condition in the **If** field of your text rule.

ℹ Note

- If you have selected the **Expression Language** as 1.0, provide the rule expression in rule expression language. For more information, see Expression Language 1.0.
- If you have selected the **Expression Language** as 2.0, model the rule expression in expression language 2.0 (DMN SFEEL). For more information, see Model a Rule Expression in Expression Language 2.0.

8. Provide the values or expressions for the result data object in the **Then** field.
9. Optional: Choose **Add Else If** to add multiple if-then conditions.
10. Optional: Choose **Validate** to check for validation errors.
11. To activate your text rule, choose **Activate**.

Model a Text Rule for Execute Operations

Context

Note

Text rule execute operations are supported only in Expression Language 2.0 using **Orchestration** ruleset execution policy.

Procedure

1. Provide a **Name**, **Label**, and **Description** for the rule.
2. Choose the **Text Rule** tab and then choose ⚙.
3. Choose **No Default Result** from the **Result** dropdown list.
4. Choose **Apply**.
5. In the **If** field, enter the rule condition.
6. Press CTRL + SPACE in the **Then** field of your text rule to view the list of operations and loop functions that can be performed on any data object of the project.
7. Choose the operation or loop function to be performed on data objects, and then provide the required input for the operation or function as per the syntax. For more information on the syntax of execute operations and loop functions, see *Functions for Text Rule Execute Operations* in Functions.

   If you want to configure loop functions, do the following:

   a. Choose **Loop Functions**.

   b. In the **Configure Loop Functions** window, from the **Function** dropdown choose **FOREACH**.

   c. From the **Vocabulary** dropdown, choose a data object of type **Table** or a rule that returns a data object of type **Table**.

   d. (Optional) In the **Where** field, provide a filter condition for retrieving the values from the **Vocabulary**.

   e. Press CTRL + SPACE in the field to configure the execute operations to be executed in a loop.

   f. Choose **Apply**.

   You can view the loop function syntax in the **Function Label** field. To edit the function, choose the function syntax in the **Then** section of the text rule.
8. (Optional) To add more result operations, choose +.
9. (Optional) Choose **Add Else If** if to add multiple if-then conditions.
10. (Optional) Choose **Validate** to check for validation errors.
11. To activate your text rule, choose **Activate**.

Model a Rule Expression in Expression Language 2.0

You can use the autosuggestion feature in Expression Language 2.0 to model a rule expression.

Prerequisites

You are familiar with Expression Language 2.0.

For more information, see Expression Language 2.0.

Context

Expression Language 2.0 lets you seamlessly model a rule expression by selecting the required entity of the rule expression from the autosuggest list. You can also use free flow typing to enter the rule expressions, which lets you type the rule expressions in the field and select the corresponding vocabulary from the autosuggest list.

Note

To include vocabulary rules in the autosuggest list, add the vocabulary rules to the corresponding ruleset vocabulary. Result data objects of the vocabulary rules should not be part of other rule conditions.

Procedure

1. To display the autosuggest list, press Ctrl + Space or start typing your rule expression.

   ### Note

   For more information on free flow typing, see Autosuggest List in Expression Language 2.0.

2. From the autosuggest list, select the data object or rule and then the relevant operators or functions if required.

   ### Note

   You can select the vocabulary only from the autosuggest list. You can select the operators or functions from the autosuggest list or type it as a part of the rule expression.

3. If your rule condition has a fixed value, provide the value using the correct syntax. You can also enter the value in the **Fixed Value** field of the autosuggest list.

   For more information on the type of fixed values and examples, see Fixed Values.

   ### Note

   If you have configured the value help for an attribute, you can select the fixed value of the attribute from its value help. To select the fixed value of the attribute from the value help, choose .

   For more information on adding value help to attributes, see Adding Value Help.

4. Optional: To use select functions, choose **Select Functions**, and perform the following steps:

   a. Select the required select function from the **Functions** dropdown list.

   b. Provide the number of values to be displayed in the **Count** field.

   ### Note

   The **Count** field is not available for the SELECT function.

c. Select a data object or a rule returning a data object of type **Table** or **Structure** from the **Vocabulary** dropdown list.

d. In the **Attributes** field, select the required attributes from the autosuggest list.

e. Choose the sorting order from the dropdown list.

f. In the **Where** field, provide a filter condition for retrieving the values from the **Vocabulary**, for performing the select function.

> **Note**
>
> Filter condition should be provided as fixed value or a data object of type **Element** against one or more attributes of the **Vocabulary**.

> **Example**
>
> `Age > 20 AND Email = Customer.Email`
>
> `Age` and `Email` are attributes of the data object selected in the **Vocabulary** field.

- The values retrieved using select functions are based on the filter condition that you provide in the **Where** field.
- Conditions entered in the **Where** field is deleted when you change the **Vocabulary**.

You can view the select function that configured in the **Function Label** field.

5. Optional: To use the aggregate functions, choose **Aggregate Functions**, and perform the following steps:

a. Select the required aggregate function from the **Function** dropdown list.

b. Select a data object or a rule returning a data object of type **Table** from the **Vocabulary** dropdown list.

> **Note**
>
> You can perform aggregate operations on select functions. To do this, use select functions on data objects of **Table** type in the **Vocabulary** field using the autosuggest list.

c. In the **Attribute** field, select the attribute on which you want to perform aggregate functions, from the autosuggest list.

> **Note**
>
> If the **Vocabulary** is based on select function, then **Attribute** of the aggregate function is the same as that of the select function, and cannot be edited.

d. In the **Where** field, provide a filter condition for retrieving the values from the **Vocabulary**, for performing the aggregate function.

> **Note**
>
> Filter condition should be provided as fixed value or a data object of type **Element** against one or more attributes of the **Vocabulary**.

> **Example**
>
> `Age > 20 AND Email = Customer.Email`
>
> `Age` and `Email` are attributes of the data object selected in the **Vocabulary** field.

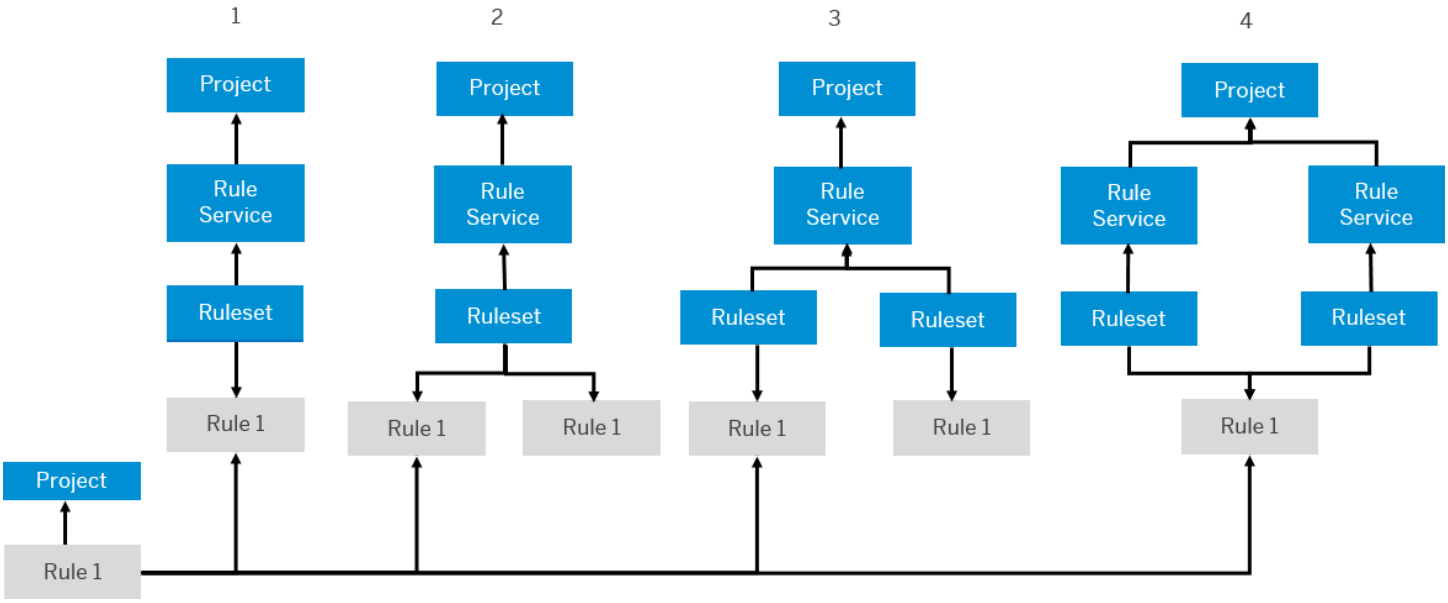
e. To group the values by an attribute, choose the attribute from the **Group by** dropdown menu.

You can view the aggregate function that you configured in the **Function Label** field.

Ruleset

A ruleset is a logical collection of rules that helps you group business logic that govern a specific function. A ruleset links a rule or group of rules to a rule service.

The following diagram shows how a ruleset is consumed:



- **Simple rule**

  A single rule is consumed by a ruleset, which is then used in a rule service.

- **Multiple rules executed by one rule service**

  Multiple rules are consumed by a ruleset, which is then used in a rule service.

- **Rulesets added in different source systems**

  Multiple rules are consumed by multiple rulesets, which is then used in a rule service.

- **One rule mapped to multiple rule services**

  Single rule is consumed by multiple rulesets, which is then used in multiple rule services. This can be consumed by a project.

Business rules capability supports two types of ruleset execution policy:

Aggregation

Aggregation execution policy creates an aggregate of the results of a set of rules. The rules are executed in the order in which they are added and this order can't be modified.

Orchestration

ℹ Note

This policy can be used only in projects that are modeled using expression language 2.0.

Orchestration execution policy is more flexible and has an order of execution of rules. You can also change the order of execution of rules. The result of one rule can be used as the input for another rule. It also supports the following features:

- Sequential execution of rules in a ruleset.
- Priority value for each ruleset for the ordered execution of rulesets.
- Ruleset vocabulary which is the specific vocabulary for each ruleset.
- Using the same data object as the input and result data object in a rule. The value of the same data object is updated on executing the rule.
- Addition of rules with different result data object in a ruleset.

Using orchestration execution policy, you can also use text rules with execute operations along with other rules to create your decision logic. You can create an ordered execution to ensure that the data objects' values are updated in a specific order. For more information about execute operations, see **Model a Text Rule for Execute Operations** in Model a Text Rule.

Related Information

Define Rulesets

Define Rulesets

You can define the set of rules to be executed when a rule service is deployed.

Prerequisites

- You have created the rule service to deploy a set of rules. For more information, see Model a Rule Service.
- You have created the rules to be associated with the rule service. For more information, see Model Rules.

Context

A set of rules can be associated to one rule service. When the rule service is deployed, the set of rules associated to it gets executed. For more information, see Ruleset.

Procedure

1. Navigate to the **Rulesets** tab of your project.
2. Provide the label, name, and description for the ruleset.
3. From the **Rule Service** dropdown list, choose the required rule service.
4. From the **Execution Policy** dropdown list, choose the execution policy.

   Note

   All the rulesets associated to a rule service should have the same **Execution Policy**.

   - If the execution policy is **Aggregation**, perform the following steps:

     a. Choose the **Rules** tab.

     b. To add rules, choose **Add Rule**.

     Note

     - To add a rule to the top of the list, choose **Insert First**.

- To add a rule in between the list, select the rule below which you want to add the rule, then choose **Insert After**.

c. Choose the required rule from the dropdown list, then choose **Save**.

> **Note**
>
> Choose the required rule from the list to navigate to the rule details page. To edit the rule, choose **Edit** in the rule details page.

- If the execution policy is **Orchestration**, perform the following steps:

a. In the **Priority** field, enter the priority value.

> **Note**
>
> Priority defines the order in which a set of rulesets is executed. You can provide a priority value between 0.01 and 99.99. The value 0.01 has the highest priority.

b. To define the ruleset vocabulary, choose the **Vocabulary** tab.

- To select the data objects that are part of the ruleset, choose in the **Data Objects Included** field.
- You can exclude the data objects that are not used in the ruleset. To select the data objects to be excluded from the ruleset, choose in the **Data Objects Excluded** field.
- To select the rules to be used in the ruleset, choose in the **Rules Included** field. For more information on vocabulary rules, see **Vocabulary Rules** in Rule.

> **Note**
>
> - You can use only those entities which are part of the ruleset vocabulary to model rules that are part of this ruleset.
> - If a rule is a part of multiple rulesets, only the common entities in the ruleset vocabularies of all the rulesets are available for modeling the rule.

c. Choose the **Rules** tab.

- Choose **Add Rule**, then select the rule to be added from the dropdown menu.

> **Note**
>
> To add a rule to the top of the list, choose **Insert First**. To add a rule in between the list, select the rule below which you want to add the rule, then choose **Insert After**.

- To change the order of execution of rules, select the rule, then choose ^ or v to move the rule up or down.

> **Note**
>
> Choose the required rule from the list to navigate to the rule details page. To edit the rule, choose **Edit** in the rule details page.

5. To check if the ruleset is free of validation errors, choose **Validate**.
6. To activate your ruleset, choose **Activate**.

Include a Project

You can include another project in your project to consume its vocabulary.

Prerequisites

The project to be included in your project has an exposed vocabulary. For more information, see Expose Data Objects and Rules.

Context

You can only consume the exposed vocabulary of the included project in your project.

Procedure

1. In the **Manage Rule Projects** application, open the **Details** tab of your project.
2. In the **Included Projects** field, choose .
3. Search for the projects to be included, either by the project name or project label, then select the required projects.

> **Note**
>
> - This search is case sensitive.
> - You can only include versioned projects. For more information, see Manage Project Versions and Revisions.
> - You can only include two projects in your projects. If the project that you include has an included project, then the included project is also counted as a project.

Results

- You can see the included projects in the **Included Projects** field in the **Details** tab.
- The exposed data objects of the included project are listed in the **Included Data Objects** section in the **Data Objects** tab.
- The exposed rules of included project are listed in the **Included Rules** section in the **Rules** tab.

Expose Data Objects and Rules

You can expose the data objects and the rules of your project so that you can use them in other projects.

Context

You reuse your project content by exposing the vocabulary of your project. The vocabulary that you expose can then be used in projects that include your project. You can only expose the data objects and the rules of your project.

Procedure

1. In the **Details** tab of your project, navigate to the **Exposed Vocabulary** section.
2. To expose data objects, perform the following steps:
    a. In the **Data Objects** field, choose .
    b. Select the data objects that you want to expose, then choose **OK**.
3. To expose rules, perform the following steps:

a. In the **Rules** field, choose .

b. Select the rules you would like to expose, then choose **OK**.

> **Note**
>
> The exposed data objects and rules are categorized as included data objects and included rules in the project that includes your project.
>
> For more information, see Include a Project.

Migrate to Expression Language 2.0

If you want to migrate your project from Expression Language 1.0 to Expression Language 2.0, you can use the migration feature.

Prerequisites

- Your project is active.

Context

You can change the expression language of your project from Expression Language 1.0 to Expression Language 2.0. Use the following procedure to change the expression language from 1.0 to 2.0.

> **Note**
>
> You can only change the expression language from 1.0 to 2.0 and not the vice versa.

For more information on the expression languages in business rules capability, see

- Expression Language 1.0
- Expression Language 2.0

Procedure

1. From **Manage Rule Projects** screen, choose your project, which is in expression language 1.0.
2. In the **Details** tab of your project, choose **Migrate**.
3. Optional: To migrate your project to Expression Language 2.0 using APIs, see */v1/migrations* under *Projects* in SAP Business Rules Service APIs.

> **Note**
>
> If you are using rule builder control in your application, set the expression language object in your code to migrate to expression language 2.0:
>
> ```
> oAstExpressionLanguage = new sap.rules.ui.services.AstExpressionLanguage();
> oRuleBuilder.setAstExpressionLanguage(oAstExpressionLanguage);
> ```
>
> For more information, see *Coding* section in:
>
> - Associating the Expression Language for Decision Tables
> - Associating the Expression Language for Text Rules

Deploy a Rule Service

You can deploy a rule service from the **Manage Rule Projects** application.

Prerequisites

- Each entity of the project is active.
- If it is a remote target runtime system, you have assigned a **System** to your project while creating it.

For more information, see Create a Project.

Procedure

1. Choose the required project from the **Manage Rule Projects** application.
2. Choose the **Rule Services** tab.
3. Choose the rule service that you want to deploy from the **Rule Services** section.

   Note

   You can only deploy the active rule services.

4. Choose **Deploy**.

   Note

   You can undeploy the rule service from the cloud runtime.

   For more information, see Rule Execution API for Cloud Foundry.

5. From **System** dropdown list, select the target runtime system to deploy the rule service.

   Note

   - You can deploy a rule service locally to cloud runtime.
   - To deploy a rule service to another target system, navigate to the **Details** tab, then change the assigned **System**. After validating the project, activate and deploy the rule service.
   - To deploy a rule service in to a different runtime, change assigned annotations at data object level, data object attribute level and rule service level.

Manage Project Versions and Revisions

You can create different versions of your project and perform different tasks on the versioned projects. You can also create a revision for a group of project versions.

Prerequisites

You have activated your project. For more information, see Create a Project.

Context

Versions are useful when you want to periodically release different versions of your project to the customer or add patch changes to the project. Each version of your project can have minor changes in technical or business logic.

ℹ Note

You can create upto ten different versions of your project.

When your project has a considerable number of changes when compared to the previous versions, you can create a new revision of your project. A revision is a group of versions. The project and its entities should be active for creating a version. You cannot edit a released version of your project.

Version has three parts where each part is a number with a maximum of six digits, separated by periods.

Example

The following table is the list of project revisions and versions maintained for a tax computation application:

Revision	Versions
TaxesProject2019	2.0.1
	2.0.0
TaxesProject2018	1.1.0
	1.0.1
	1.0.0

In TaxesProject2019, 2.0.0 is the first or the main version of the project. 2.0.1 project version includes the patch changes added to the project version 2.0.0, by a business user. In TaxesProject2018, 1.0.0 is the first or main version of the project and 1.1.0 version includes the changes released by a developer for the business users.

→ Recommendation

We recommend that you represent the changes to the rule expression or logic by a business user, as a patch change.

You can perform the following tasks on the versioned projects:

- View the version history
- Deploy the rule services of other versions of your project
- Copy the content of a versioned project to a project that you are working on
- Export the versioned project to your local system or a transport system
- Delete a version of your project

Procedure

1. To create a version of your project, perform the following steps:
    a. From the **Manage Rule Projects** application, choose your project, then choose **Release Version**.
    b. Provide the following information in the new window:

Field	Description
Version	Version number with three parts, with each part separated by periods.

Field	Description
Revision	Revision information that is used to group different versions of your project. It should be provided in alphanumeric characters.
Description	Version description.

c. Choose **Release**.

2. To view the version history of your project, perform the following steps:

   a. From the **Manage Rule Projects** application, select your project.

   b. Choose **History**.

3. To deploy the rule service of one of the versions of your project, perform the following steps:

   a. Choose the version of your project from **History**.

   b. Navigate to the **Rule Services** tab, then select the rule service that you want to deploy.

   c. Choose **Deploy**.

4. To copy the content of the versioned project to the project that you are working on, perform the following steps:

   a. From **Manage Rule Projects** application, select the project that you are working on.

   b. Choose **History**, then choose the version of the project that you want to copy.

   c. Choose **Copy to Draft**.

   Note

   It overwrites the content of the project that you are working on, with the content of the versioned project and opens the project in **Edit** mode.

5. To export the versioned project to your local system, perform the following steps:then

   a. From **History**, select the version of the project that you want to export.

   b. Choose ⤓ then choose **Download Project to File System**.

   The versioned project is exported as a .zip file to your local system.

6. To export the versioned project to a transport system, perform the following steps:

   a. From **History**, select the version of the project that you want to export.

   b. Choose ⤓ then choose **Export Project to Transport System**.

   c. Select **System** and **Transport Type** from the dropdown lists.

   Note

   - If the transport type is **Workbench**, then the package gets enabled and it is mandatory.
   - If the transport type is **Customizing**, then the package is not required.

   d. (Optional) Provide the **Owner** name, then choose **GO**.

   e. (Optional) Select the **Request** from the search resuts.

   Note

   If the owner name is not provided, then it fetches the user name, which is provided in the managed system destination.

   f. Choose **Export**.

Note

To delete a version of your project, see Delete a Version of a Project.

Delete a Version of a Project

Delete a version of your project using **Manage Rule Projects** application or API.

Procedure

1. To delete a version of a project using the **Manage Rule Projects** application, do the following:
    a. Open the **Manage Rule Projects** application.
    b. From the **Manage Projects** window, select the project of which you want to delete a version.
    c. Choose **History**.

    You will be able to view the list of versions of the project.

    d. From the list of versions, select the version that you want to delete and then choose *Delete Version*.
2. To delete a version of a project using API, do the following:
    a. Navigate to Business Rules on SAP API Business Hub.
    b. Select **Rule Authoring API for Cloud Foundry** tile.
    c. Under **Projects** section, execute the API `/v1/projects/{id}/versions/{version}` from SAP API Business Hub or any REST client of your choice.

Export a Project

You can export a project to your local system from the **Manage Rule Projects** application.

Procedure

1. From **Manage Rule Projects** application, select the project that you want to export.
2. Choose to download the project to your local file system.

The project .zip file is downloaded to your local system with project name as its file name.

Note

- The .zip file includes information about the exported project and its entities, such as data objects, rule services, rules, and rulesets.
- The entity file name has the following format: `<entity name>.br<entity type>`. For example, if a project contains the data object "customer," then the file is exported as `customer.brdataobject`.

Import a Project

You can import a project from your local system or a transport system to the **Manage Rule Projects** application.

Prerequisites

- If you are importing the project from SAP API Business Hub or a transport system, you have configured the system. For more information, see Configure a System.

- You have translated the labels and descriptions of all the entities to the user language. For more information, see Translate Labels and Descriptions.

Context

You can import a template project into your application and customize the project as per your requirements. The project can be imported from your local file system or a transport system.

You can also import a project from SAP API Busines Hub into your application. The digital content package in SAP API Business Hub contains multiple projects, which can be consumed in your application.

Procedure

1. Choose the **Manage Rule Projects** application.
2. Choose ↥.
3. To import a project from your local file system, choose **Upload Project from File System**, then perform the following steps:
    a. Choose **Browse** to search for the file in your local system.

    > **Note**
    > You can import only those projects, that are exported in zip format from the manage rules project application.

    b. Select the file, then choose **Import**.
4. To import the project from a transport system, choose **Import Project from Transport System**, then perform the following steps:
    a. Select **System** that you have configured from the dropdown list.
    b. Select **Transport Type** from the dropdown list.
    c. Optional: Provide the **Project Name**, which you want to import, then choose **GO**.

    > **Note**
    > - If the project name is not provided, it will list all the projects of that particular transport type, then you can select it from the listed projects.
    > - You can also append * with the project name while searching to find the list of similar project names. For example, consider there are projects like **Test1**, **Test2**. While searching with this format **Tes***, it will provide you all the projects which starts with **Tes** in the search results.

    d. Select the project, then choose **Import**.
5. To import the project from SAP API Business Hub, choose **Import Project from API Business Hub**, then perform the following steps:
    a. Select **System** that you have configured from the dropdown list.
    b. In the **Content Package Name** field, provide the name of the digital content package from which you want to import the project, then choose **GO**.

    > **Note**
    > **Content Package Name** is case sensitive and do not support special characters.

    c. Choose the required project, then choose **Import**.

Translate Labels and Descriptions

Translate the labels and descriptions of your project and its entities into the user language before providing the project to the business user.

Context

An application developer should manually translate the label and description of the project and its entities like data objects, rules, rulesets, and rule services before providing the project to the business user who uses a different language in their locale. It is important to translate the labels of all the entities of the project since the label of the entities are displayed in the autosuggest list and breadcrumb navigation. This lets the user model rules in their language.

Procedure

1. Export the project to your local system. For more information, see Export a Project.

   The project is saved as a zip file.

2. Extract the project file with the extension `.brproject` from the zip file.
3. Open the project file to edit the project code.
4. Create new key value pairs, `Language` and `Text`, for `Label` and `Description` in business user language.

Note

Provide the `Language` in ISO 639-1 language code.

Example

The following is the code snippet for project details in English:

```
"Project" : [ {
    "Id" : "c753ebb01ae149f5bc39351d4d832d61",
    "Name" : "Test_Project",
    "Description" : [ {
      "Language" : "",
      "Text" : "Test Project"          //This is the default text for Description.
    }, {
      "Language" : "en",
      "Text" : "Test Project"
    }],
        "Label" : [ {
      "Language" : "",
      "Text" : "Project"                    //This is the default text for Label.
    }, {
      "Language" : "en",
      "Text" : "Project"
        } ]
```

If the label and description is to be translated into Spanish, provide the following language code and the translated text:

```
"Project" : [ {
    "Id" : "c753ebb01ae149f5bc39351d4d832d61",
    "Name" : "Test_UA",
    "Description" : [ {
      "Language" : "",
      "Text" : "Test Project"
```

```
}, {
  "Language" : "en",
  "Text" : "Test Project"
}, {
  "Language" : "es",
  "Text" : "Proyecto de prueba"
} ],
    "Label" : [ {
  "Language" : "",
  "Text" : "Project"
}, {
  "Language" : "en",
  "Text" : "Project"
},{
  "Language" : "es",
  "Text" : "Proyecto"
    }
```

5. Save and zip the project file, then import the file into the **Manage Rule Projects** application in the user locale.
6. Activate each entity of the project by choosing **Activate** in their respective tabs, then activate the project. You can also activate the project via API.

Using Business Rules Capability APIs

The REST-based API allows a tight integration of tasks on SAP BTP with business rules capability .

Business Rules Capability exposes two types of APIs to address different use cases.

- REST-based APIs
- OData-based APIs

The REST-based APIs are the main design and runtime APIs that lets you invoke the business rules capability into your own custom application and allows the execution of tasks when the service is not bound to any UI applications.

The OData-based APIs allows users to invoke the business rules capability into the SAP UI5 applications that are developed using the rule builder control. These APIs are used to perform tasks when the services that are bound to UI applications.

Access the API Documentation

See the Business Rules APIs in SAP API Business Hub.

Authentication

Clients must authenticate to use the business rules capability APIs. The following authentication types are supported:

- OAuth2 (client credentials)

To ensure optimal operation of the service, REST API execution is subject to resource limits, for example, regarding the number of requests per second. If the limits are exceeded, API calls return HTTP status 429 (“Too many requests”). The client should then reduce the number of calls.

ℹ Note

For all OAuth2-based business rules APIs, you do not need to specify a CSRF token. This is because the OAuth2 flows already have CSRF protection when used without intermediaries. However, when business rules API requests are routed through an application router, you must apply CSRF token mechanisms. Otherwise, the CSRF protection is lost. Therefore, do not turn off the `csrfProtection` setting of routes in the application router that use xsuaa as `authenticationType`.

Related Information

Determine Service Configuration Parameters
Access Business Rules Capability APIs Using OAuth 2.0 Authentication (Client Credentials)

Determine Service Configuration Parameters

You need to determine the OAuth client credentials and endpoint URLs. These are the basic service configuration parameters required to consume the business rules capability APIs.

Prerequisites

- You have created a service instance of the business rules service instance.

  For more information, see Create a Service Instance of Business Rules Capability Using the Cockpit.

- You have bound the service instance to your Cloud Foundry application, or you have created a service key.

Procedure

1. Navigate to your Cloud Foundry subaccount.
2. In the navigation area, choose **Spaces** and navigate to the space in which you have created a service instance.
3. Choose **Services** **Instances and Subscriptions**.
4. Choose the Business Rules instance and choose **Service Keys**.
5. From the json file of the service key, copy the following service configuration parameters:

Parameter Name	Description	Example
`clientid`	Client ID for OAuth 2.0	`sb-clone-34ce1dfd-5076-4fc4-ba38-e5166f455dbf!b73732|bpmrulebroker!b2466`
`clientsecret`	Client secret for OAuth 2.0	`c7c567d9-feec-430e-a4c5-5ad145e9d235$yOGFEelMPh6NnDfkZ8Owbt-FRzLxFVpYlMhnu7NWXq4=`
`url`	Authentication base URL for OAuth2	`https://<subdomain>.authentication.<region host>.hana.ondemand.com`
`rule_repository_url` and `rule_runtime_url`	Endpoint URLs: Repository and runtime URLs to access the APIs.	`https://bpmrulerepository.cfapps.<region>.hana.ondemand.com` and `https://bpmruntime.cfapps.<region>.hana.ondemand.com`

Access Business Rules Capability APIs Using OAuth 2.0 Authentication (Client Credentials)

You can access the business rules capability APIs using an access token from OAuth 2.0 authorization server.

Prerequisites

You have determined the service configuration parameters. For more information, see Determine Service Configuration Parameters.

Procedure

1. To request an access token, send the following POST request through a REST client.

URL	{url}/oauth/token
Method	POST
Authorization	Basic authentication Username: clientid Password: clientsecret
Headers	Content-Type: application/x-www-form-urlencoded
Body	grant_type: client_credentials response_type: token

2. Copy the access token from the HTTP response.
3. Perform a call to the business rules capability API by sending the access token as its header.
    - **Header Name**: Authorization
    - **Header Value**: Bearer <access token>

Note

For runtime APIs, use the following URL in the API call request:

{rule_runtime_url}/rules-service/rest/{API}

For repository APIs, use the following URL in the API call request:

{rule_repository_url}/rules-service/rest/{API}

Migrate Business Rules Execution APIs to the New Version

This section describes the process of migrating your applications from v1 version of business rules execution APIs to the v2 version and provides you with a general overview of the changes in the v2 APIs.

Context

Version v2 of the business rules execution APIs is now available for consumption and comes with several improvements. This section describes the relevant changes in detail and offers code examples that demonstrate the steps that are necessary for a successful and seamless migration. The following are the main improvements in v2 APIs:

- **API endpoint changes:** The API endpoint path is now fixed for all invocations of v2 API. Input parameters are passed as a part of the API payload and not the endpoint.
- **API payload changes**: The v1 API JSON payload has a custom structure and all the input vocabularies are related to each other if there are multiple inputs or references. You cannot model a rule service that has multiple vocabularies with no relation to each other. The relation between inputs is resolved by the runtime in v1 APIs. The v2 APIs now have a

standard JSON structure and the signature is similar to that of the rule service model. Rule service in v2 API can have multiple inputs or references that are independent of each other and also lets you use an output of type **Table**.

- **Versioning support**: The v2 APIs supports the execution of versioned rules.
- **Changes in API execution behavior**: You can use the same type of data object as the input as well as the result. You can also include multiple independent inputs as a part of the payload of the API. If a rule service returns multiple results, then the result data objects should be of **Table** type. If the **Hit Policy** of decision table is **All Match**, then the rule engine returns all the matching results as a table of records.

Since new features are available only in v2 APIs, we strongly recommend to move your applications to the new version.

ℹ Note

Version v1 of the rule execution APIs is deprecated and is supported only for backward compatibility. Version v1 will be supported until May 2020.

Procedure

1. Change the API endpoints and JSON payload of v1 APIs as per that of v2 APIs. For more information see, Business Rules APIs.

Example

This is a comparison of the version 1 and version 2 APIs used to invoke a rule service.

V1 version:

API endpoint: `/v1/rule-services/java/{projectName}/{ruleserviceName}`

Input JSON payload example:

```
[
  {
    "__type__": "Employee",
    "id": "LORIN",
    "name": "John",
    "age": "32"
  },
  {
    "__type__": "Employer",
    "id": "001",
    "name": "SAP"
  }
]
```

Response:

Code: 200 (Successful invocation of rule service)

```
[
  {
    "__type__": "EmployeeDiscount",
    "id": "1090",
    "discount": 3.25
  }
]
```

V2 version:

API endpoint: `/v2/workingset-rule-services`

JSON payload example:

```
{
  "RuleServiceId": "0207ff52b2954a84a9d50fa593812487",
    "Vocabulary": [
    {
      "Employee": {
        "id": "LORIN",
        "name": "John",
        "age": "32"
      },
      "Employer": {
        "id": "001",
        "name": "SAP"
      }
    }
  ]
}
```

Response:

Code: 200 (Successful invocation of rule service)

```
{
  "Result": [
    {
      "EmployeeDiscount": {
        "name": "John",
        "discount": "10"
      }
    }
  ]
}
```

2. If you are modeling a decision table, with the **Hit Policy** as **All Match**, then choose a result data object of **Table** type. To do this, perform the following steps:

    a. Navigate to the **Data Objects** tab, then create a data object of **Table** type. For more information, see Model Data Objects.

    b. Choose the data object of **Structure** type from the **Reference Data Object** dropdown list.

    c. Navigate to the **Rules** tab, and select your decision table.

    d. Set the new data object of **Table** type as the **Result Data Object** of the decision table.

ℹ Note

The decision table values are reset during this process and you have to reenter the values.

3. If a rule service returns multiple results, choose a result data object of **Table** type. To do this, perform the following steps:

    a. Navigate to the **Data Objects** tab, then create a data object of **Table** type. For more information, see Model Data Objects.

    b. Choose the data object of **Structure** type from the **Reference Data Object** dropdown list.

c. Navigate to the **Rule Service** tab.

> **Note**
>
> If there is an existing entry for **Result**, delete the entry.

d. Choose **+**, and from the dropdown list, select the **Data Object** of **Table** type.

e. Select **Usage** as **Result** from the dropdown list.

Build a SAP Fiori Application to Author a Business Rule

Build Fiori applications to consume the business rules capability, using its capabilities to assist you in writing business rules.

Context

- Build a Fiori application that can consume business rules capability in a standalone application. You don't need to build a transactional integrity between the service's create/read/update/delete (CRUD) transactions, and the consuming Fiori application transactions. The service's OData service can be consumed directly from its user interface.
- Build a Fiori application that is embedded with the business rules capability. In this case, you need to build transactional integrity between the service's CRUD transactions and the consuming Fiori application transactions. You must also include the OData model into the hosting Fiori application OData service.

Procedure

1. Model the back-end access in neo-app.json.

```
{
   "path": "/sap/opu/odata",
   "target": {
     "type": "destination",
     "name": "<your backend destination>",
     "entryPath": "/sap/opu/odata"
   },
   "description": "backend destination for rules access "
 },
 {
   "path": "/sap/bc",
   "target": {
     "type": "destination",
     "name": "<your backend destination>",
     "entryPath": "/sap/bc"
   },
   "description": "backend destination for personalization data"
 }
```

2. Add the **RuleBuilder** control to **Application View**.

   a. Create a **DecisionTable** control.

   b. Enable editing settings.

   c. In the OData model, set the map control editability to **IsDraft** attribute.

   Provide the following code in the **Application View**:

```
xmlns:rules="sap.rules.ui"

<rules:RuleBuilder id="ruleBuilder"
    types="DecisionTable"
    editable="{path:'IsDraft', formatter: '.formatterIsDraft' }">
</rules:RuleBuilder>
```

3. Create an instance of the Business Rules Capability model in **Application Controller**.

ℹ Note

Obtain sRuleId and sVersion is by triggering a GET call in the OData model. In case the access point is a given project, and user selects a rule within a project. You use the association **Rules** from **Projects** entity set. It returns rules with the latest version.

sRuleId and sVersion should be obtained by triggering a get call in the oData Model. In the case the access point is a given project, and user selects a Rule within a project, use association "Rules" from "Projects" entity set. It returns Rules with the latest version.

If the **Rule Id** is known, filter the rule by **Id** and **ValueTo** = null. Refer to the following example:

Sample Code

```
//Global in controller
var oVocabularyJson = {};

// onInit
var sRuleId = "<Id>";
var sVersion = "<Version>";

var oRuleBuilder = this.byId("ruleBuilder");

//Set style class in the containing page. Pending control correction
var styleClass = !sap.ui.Device.support.touch ? "sapUiSizeCompact" : "sapUiSizeCozy";
$('body').addClass(styleClass);

//Enable changing rule settings
var oDTConfig = new sap.rules.ui.DecisionTableConfiguration({ enableSettings: true });
oDecisionTable.setDecisionTableConfiguration(oDTConfig);

/////In case the Rule model differs to model assigned to the view ////////////////////////
var oRuleModel = null;

//In case of Fiori template get model from owner component
oRuleModel = this.getOwnerComponent().getModel();

//In case of non existing component
//oRuleModel = new sap.ui.model.odata.v2.ODataModel("/sap/opu/odata/SAP/rule_srv/")
oRuleModel.setDefaultBindingMode(sap.ui.model.BindingMode.TwoWay);

oRuleBuilder.setModel(oRuleModel);
////////////////////////////////////////////////////////////////////////////////////////////

var oVocabularyModel = new sap.ui.model.odata.v2.ODataModel("/sap/opu/odata/SAP/vocabulary_
var sRuleKey = oRuleModel.createKey("Rules", { Id: sRuleId, Version: sVersion });
this.sRulePath = "/Rules(" + sRuleKey + ")";
var sVocabularyKey = oVocabularyModel.createKey("Vocabularies", { Id: sRuleId});
var sVocabularyPath = "/Vocabularies(" + sVocabularyKey + ")";
oDecisionTable.setBindingContextPath(this.sRulePath);

var readVocabularySuccessHandler = function(oInDecisionTable, oDataVocabulary) {
    var oExpressionLanguage = new sap.rules.ui.services.ExpressionLanguage();      //For ex
    oExpressionLanguage.setData(oDataVocabulary);
    oVocabularyJson = oDataVocabulary;
```

```
            oInDecisionTable.setExpressionLanguage(oExpressionLanguage);
        };
        oVocabularyModel.read(sVocabularyPath, {
            urlParameters: {
                "$expand": "DataObjects/Associations,DataObjects/Attributes,ValueSources,Rules"
            },
            success: function(data) {
                  if (!oExpressionLanguage) {
                     oExpressionLanguage = new sap.rules.ui.services.ExpressionLanguage();   ////Fo
                     oRuleBuilder.setExpressionLanguage(oExpressionLanguage);
                }
                oExpressionLanguage.setData(data);
                oExpressionLanguage.setBindingContextPath("/Vocabularies(\'" + <contextId> + "\')")
                oExpressionLanguage.setModel(that.oVocabularyModel);
                return readVocabularySuccessHandler(oDecisionTable, data);

            },
            error: function(oError) {
                sap.m.MessageBox.error(oError);
            }
        });

        //Handle isDraft formatter
        formatterIsDraft: function(d) {
            return d;
        },
        formatterNotIsDraft: function(d) {
            return !d;
        },
        }
```

4. Model business rules capability function import handlers.

   Use the following **Function Imports** to support the editing life cycle of the business rules service:

   - **EditRule**: Creates a draft version of the rule for the current user, **IsDraft** becomes true.
   - **SaveRule**: Saves draft changes into core persistency and deletes draft, **IsDraft** becomes false.
   - **DeleteRuleDraft**: deletes draft changes, **IsDraft** becomes false.
   - **CheckRule**: Checks rule using the back-end XS parser. Can be used for draft and nondraft rules.
   - **ActivateRule**: If **isDraft**= `true`, it internally calls **SaveRule** and activates in core persistency, then **IsDraft** becomes `false`.
   - **DeleteRule**: Deletes the rule from core persistency, all versions of the rule are deleted
   - **SetRuleResultDataObject**: Set a valid result data object in the rule. Add columns is set to true by default. Rule must be in draft mode.

   ### ℹ Note

   This functionality is provided by **RuleBuilder** control in later releases.

   ### Sample Code

In Application View:

```
<Toolbar id="actionsToolbar">
    <Button id="editButton" icon="sap-icon://edit" press="onEditPress" ariaLabelledBy
    <Button id="checkButton" icon="sap-icon://check-availability" press="onCheckPress
    <Button id="saveButton" icon="sap-icon://save" press="onSavePress" ariaLabelledBy
    <Button id="activateButton" icon="sap-icon://activate" press="onActivatePress" ar
    <Button id="cancelButton" icon="sap-icon://sys-cancel" press="onCancelPress" aria
    <Button id="deleteButton" icon="sap-icon://delete" press="onDeletePress" ariaLabe
</Toolbar>
```

Sample Code

In Application Controller:

```
setResultDataObject: function() {
    var oRuleBuilder = this.byId("ruleBuilder");
    var oRuleModel = oRuleBuilder.getModel();
    var sRuleId = oRuleBuilder.getBindingContext().getProperty("Id");
    var sRulePath = oRuleBuilder.getBindingContextPath();
    oRuleModel.read(sRulePath, {
        success: function() {
            var sResultDataObjectIdProp = sRulePath + "/ResultDataObjectId";
            var sResultDataObjectId = oRuleModel.getProperty(sResultDataObjectIdProp)
            //If no result data object id provided, take default from Vocabulary
            if (!sResultDataObjectId && oVocabularyJson.DataObjects && oVocabularyJso
                oVocabularyJson.DataObjects.results.forEach(function(oDataObject) {
                    if (oDataObject.Usage === "RESULT" && !sResultDataObjectId) {
                        sResultDataObjectId = oDataObject.Id;
                    }
                });
                if (sResultDataObjectId) {
                    oRuleModel.callFunction("/SetRuleResultDataObject", {
                        method: "POST",
                        urlParameters: {
                            "RuleId": sRuleId,
                            "ResultDataObjectId": sResultDataObjectId
                        },
                        success: function(oRuleModel2, oResponseData2) {
                            oRuleModel.refresh();
                        },
                        error: function(oError) {
                            sap.m.MessageBox.error(oError);
                        }
                    });
                }
            }
        }
    });
},

handleActionPress: function(oFIParam) {
    var hasResponseErrors = function(oResponseData) {
        var sError;
```

```
                if (oResponseData.__batchResponses) {
                    oResponseData.__batchResponses.forEach(function(oResponse) {
                        if (oResponse.response) {
                            var oJsonMessage = JSON.parse(oResponse.response.body);
                            if (oJsonMessage.error) {
                                sError = sError + oJsonMessage.error.message.value + " (" + o
                            }
                        }
                    });
                }
                return sError;
            };
            var performFunctionImport = function(oResponseData, oInFIParam) {
                var sRuleId = oInFIParam.ruleBuilder.getBindingContext().getProperty("Id");
                if (!oResponseData || !hasResponseErrors(oResponseData)) {
                    oInFIParam.ruleBuilder.getModel().callFunction("/" + oInFIParam.name, {
                        method: oInFIParam.method,
                        urlParameters: {
                            "RuleId": sRuleId
                        },
                        success: oInFIParam.success,
                        error: function(oError) {
                            sap.m.MessageBox.error(oError);
                        }
                    });
                }
            };
            if (oFIParam.ruleBuilder.getModel().hasPendingChanges()) {
                oFIParam.ruleBuilder.getModel().submitChanges({
                    success: function(oResponseData) {
                        return performFunctionImport(oResponseData, oFIParam);
                    },
                    error: function(oError) {
                        sap.m.MessageBox.error(oError);
                    }
                });
            } else {
                return performFunctionImport(null, oFIParam);
            }
        },
        onActivatePress: function() {
            var oFIParam = {
                name: "ActivateRule",
                ruleBuilder: this.byId("ruleBuilder"),
                method: "POST",
                success: function(oResponseData) {
                    sap.m.MessageBox.success("Rule " + oResponseData.Id + " activated success

                    //After activation (if version management enabled and save triggered ) ve
                    var oRuleModel = this.byId("ruleBuilder").getModel();
                    var sRuleKey = oRuleModel.createKey("Rules", {
                        Id: oResponseData.Id,
                        Version: oResponseData.Version
                    });
                    this.byId("ruleBuilder").setBindingContextPath("/" + sRuleKey);
```

```
            }.bind(this)
        };
        this.handleActionPress(oFIParam);
    },
    onSavePress: function() {
        var oFIParam = {
            name: "SaveRule",
            ruleBuilder: this.byId("ruleBuilder"),
            method: "POST",
            success: function(oResponseData) {
                sap.m.MessageBox.success("Rule " + oResponseData.Id + " saved successfull
                //After save (if version management enabled) version may increase, then r
                var oRuleModel = this.byId("ruleBuilder").getModel();
                var sRuleKey = oRuleModel.createKey("Rules", {
                    Id: oResponseData.Id,
                    Version: oResponseData.Version
                });
                this.byId("ruleBuilder").setBindingContextPath("/" + sRuleKey);
            }.bind(this)
        };
        this.handleActionPress(oFIParam);
    },
    onEditPress: function() {
        var oFIParam = {
            name: "EditRule",
            ruleBuilder: this.byId("ruleBuilder"),
            method: "POST",
            success: function(oResponseData) {
                var oRuleModel = this.byId("ruleBuilder").getModel();
                //Only in case the rule is assigned to a function, it will set the single
                this.setResultDataObject();
                ////////////////////////
                oRuleModel.refresh(true);
            }.bind(this)
        };
        this.handleActionPress(oFIParam);
    },
    onCancelPress: function() {
        var oFIParam = {
            name: "DeleteRuleDraft",
            ruleBuilder: this.byId("ruleBuilder"),
            method: "POST",
            success: function(oResponseData) {
                var oRuleModel = this.byId("ruleBuilder").getModel();
                oRuleModel.resetChanges();
                oRuleModel.refresh(true, true);
            }.bind(this)
        };
        var handleActionPress = this.handleActionPress.bind(this);
        var oCancelDialog = new Dialog();
        oCancelDialog.setTitle("Cancel Rule Changes");
        var oText = new sap.ui.commons.TextView({
            text: " Do you really want to cancel your changes?"
        });
        oCancelDialog.addContent(oText);
```

```
            oCancelDialog.addButton(new sap.m.Button({
                text: "Yes",
                press: function() {
                    oCancelDialog.close();
                    handleActionPress(oFIParam);
                }
            }));
            oCancelDialog.addButton(new sap.m.Button({
                text: "No",
                press: function() {
                    oCancelDialog.close();
                }
            }));
            oCancelDialog.open();
        },
        onDeletePress: function() {
            var oFIParam = {
                name: "DeleteRule",
                ruleBuilder: this.byId("ruleBuilder"),
                method: "POST",
                success: function(oResponseData) {
                    sap.m.MessageBox.success("Rule " + oResponseData.Id + " succesfully delet
                }
            };
            var handleActionPress = this.handleActionPress.bind(this);
            var oDeleteDialog = new Dialog();
            oDeleteDialog.setTitle("Delete Rule");
            var oText = new sap.ui.commons.TextView({
                text: " Do you really want to delete this rule?"
            });
            oDeleteDialog.addContent(oText);
            oDeleteDialog.addButton(new sap.m.Button({
                text: "Yes",
                press: function() {
                    oDeleteDialog.close();
                    handleActionPress(oFIParam);
                }
            }));
            oDeleteDialog.addButton(new sap.m.Button({
                text: "No",
                press: function() {
                    oDeleteDialog.close();
                }
            }));
            oDeleteDialog.open();
        },
        onCheckPress: function() {
            var oFIParam = {
                name: "CheckRule",
                ruleBuilder: this.byId("ruleBuilder"),
                method: "GET",
                success: function(oResponseData) {
                    sap.m.MessageBox.success("Rule " + oResponseData.Id + " is consistent");
                }
            };
```

```
                this.handleActionPress(oFIParam);
            },
```

Integration

This section summarizes the tasks that need to be performed when integrating the business rules capability with another system.

For more information, see:

- Integrating with SAP S/4HANA On-Premise
- Integrating with SAP S/4HANA Cloud
- Integrating with SAP HANA

Integrating with SAP S/4HANA On-Premise

Business Rules Capability can be integrated with an SAP S/4HANA on-premise system and it is supported from version **SAP S/4HANA On-Premise 1809**.

Business Rules Framework (BRFplus) is the component of SAP S/4HANA that acts as an interface for communication between Business Rules Capability and the SAP S/4HANA system. You can maintain rules in Business Rules Capability and use them in BRFplus via remote calls.

For more information, see Business Rule Framework plus (BRFplus).

The tasks in this section describe how to configure Business Rules Capability to allow inbound and outbound communication with the SAP S/4HANA system.

- Inbound Communication
- Outbound Communication

Inbound Communication

You can configure business rules capability to allow inbound communication.

Inbound communication is the communication from business rules capability to the SAP S/4HANA system. OData services for communication reside in the SAP S/4HANA system and are invoked by business rules capability. You can invoke these services by configuring the destination and the managed system.

For more information, see the following topics:

- Configure Destination for SAP S/4HANA On-Premise System
- Configure a System

Configure Destination for SAP S/4HANA On-Premise System

You can create a destination in your subaccount to connect with the SAP S/4HANA system.

Prerequisites

The cloud connector system is connected to your subaccount to access the on-premise system's URL.

For more information, see Cloud Connector

Procedure

1. Log on to the SAP BTP cockpit.
2. Navigate to the subaccount that you want to use for deployment.
3. In the navigation area, choose **Connectivity** **Destination**.
4. Select **New Destination**.

   For more information, see Create HTTP Destinations
5. Provide the following information to create a new destination.

Option	Description
Name	<DEST_NAME>
Type	HTTP
Description	Destination description
URL	SAP S/4HANA system URL (that uses the HTTP protocol instead of HTTPS)
Proxy Type	On-premise
Authentication	Basic
User	User who is assigned S_DEVELOP and S_FDT_CP roles
Password	User password

6. Optional: Create an additional property that uses the name **sap_client** and has a value of **<S/4HANA Client ID>**.

   ### Note
   If you don't create an **sap_client** property, data is fetched from the default client of the SAP S/4HANA system.

7. Choose **Save**.

Outbound Communication

You can configure business rules capability to allow outbound communication.

Outbound communication services reside in business rules capability and are invoked by SAP S/4HANA. You need the following basic service configuration parameters to configure the OAuth client in the SAP S/4HANA system.

- **clientid**
- **clientsecret**
- **url**

For more information, see

- Determine Service Configuration Parameters
- Configuring OAuth Client.

Integrating with SAP S/4HANA Cloud

You can integrate business rules capability with an SAP S/4HANA Cloud system by configuring the communication between the two systems.

Prerequisites

- You have subscribed to the business rules capability. For more information, see Initial Setup.
- Add the key user of SAP S/4HANA Cloud as a member of the subaccount in the Cloud Foundry environment. For more information, see Add Members to Your Global Account.

Context

You can automate decision making processes in your SAP S/4HANA Cloud applications using business rules capability. There are two types of communication in this integration scenario:

- **Outbound communication**: The communication flow is from SAP S/4 HANA Cloud to business rules capability. You configure outbound communication mainly in S/4HANA Cloud. The service key parameters of the service instance of business rules capability from the SAP BTP cockpit are required for the configuration.
- **Inbound communication**: The communication flow is from business rules capability to SAP S/4HANA Cloud. The configuration for inbound communication is set up mainly in SAP BTP cockpit. Communication arrangement and the information of key user of S/4HANA Cloud are required for the configuration.

  To integrate and enable communication between the two systems, you need to configure both outbound and inbound communication.

Procedure

1. Configure inbound communication in SAP S/4HANA Cloud and business rules capability. For more information, see Configure Inbound Communication.
2. Configure outbound communication in business rules capability and SAP S/4HANA Cloud. For more information, see Configure Outbound Communication.

Configure Inbound Communication

You should ensure that the configurations for inbound communication are done in business rules capability.

Prerequisites

- A communication user is created for SAP S/4HANA Cloud key user and you have noted the **Username** and **Password**. For more information, see How to Create Communication Users.
- A communication arrangement of type **SAP_COM_0235** is enabled in SAP S/4HANA Cloud. For more information, see How to Create a Communication Arrangement.
- You have access to the **Manage Rule Projects** application. For more information, see Access the Manage Rule Projects Application.

Context

You should configure the destination for SAP S/4HANA Cloud in SAP BTP cockpit. Using this configured destination, configure the managed system in the **Manage Rule Projects** application to allow consumption of business rules capability in SAP S/4HANA Cloud.

Procedure

1. Configure destination for SAP S/4HANA Cloud in SAP BTP cockpit.
    a. Log on to SAP BTP cockpit.
    b. Navigate to the subaccount that you want to use for deployment.
    c. In the navigation area, choose **Connectivity** **Destination**.
    d. Select **New Destination**. For more information on creating destinations, see Create HTTP Destinations.
    e. Provide the following information:

Name	Destination name
Type	HTTP
Description	Description of the destination
URL	Host and port of S/4HANA Cloud system
Proxy Type	Internet
Authentication	BasicAuthentication
User	**Username** of the communication user created in the SAP S/4HANA Cloud system.
Password	**Password** of the communication user created in the SAP S/4HANA Cloud system.

    f. Choose **Save**.
2. Using the destination, configure the SAP S/4HANA Cloud as the managed system in the **Manage Rule Projects** application. For more information, see Configure a System.

Configure Outbound Communication

Configure the SAP S/4HANA Cloud to initiate outbound communication with business rules capability.

Prerequisites

You've created a service instance of business rules and created the service keys in SAP BTP cockpit. For more information, see Create a Service Instance of Business Rules Capability Using the Cockpit.

Context

To configure the communication settings in SAP S/4HANA Cloud, you need the service key parameters of the business rules service instance.

Procedure

1. Copy the parameters required to configure communication system in SAP S/4HANA Cloud from the service key of the business rules service instance. To copy the parameters, perform the following steps:
    a. Log on to SAP BTP cockpit.
    b. In the navigation area, choose **Spaces** and navigate to the space in which you have created a service instance.
    c. Choose **Services** **Service Instances**, then choose your business rules service instance.
    d. From the JSON file, copy the following parameters:

- clientid
- clientsecret
- rule_repository_url
- url

2. In SAP S/4HANA Cloud, do the following:
    a. In your S/4HANA Fiori Launchpad, choose the Communication Management tab.
    b. Choose the Communication Systems tile and then choose New.
    c. In the dialog, provide the System ID and System Name and then choose Create.
    d. In the Technical Data section, in the Hostname field, provide the rule_repository_url without https://.
    e. In OAuth 2.0 settings, in the Auth Endpoint field, provide the url without https://. In the Token Endpoint field, provide the url without https:// and suffixed with /oauth/token.
    f. In the Users for Outbound Communication section, choose + to add a new user and provide the following details:

Field	Input
Authentication Method	Select OAuth 2.0
OAuth 2.0 Client ID	clientid from the JSON
Client Secret	clientsecret from the JSON

    g. Choose Create and then choose Save.

    For more information, see How to Create Communication Systems.

3. In SAP S/4HANA Cloud, enable communication arrangement of type SAP_COM_0236. For more information, see How to Create a Communication Arrangement.

Integrating with SAP HANA

You can integrate business rules capability with SAP HANA System.

You need to configure business rules capability to allow communication with SAP HANA system. The rules that are modeled in business rules capability can be consumed in SAP HANA system via the SAP HANA Rules Framework.

For more information, see Configure Integration with Business Rules.

Related Information

Configure Destination for SAP HANA System
Configure a System

Configure Destination for SAP HANA System

You should configure the destination on SAP BTP cockpit to ensure communication between business rules capability and SAP HANA system.

Prerequisites

- You have the credentials (client ID, secret client), XS UAA URL, and the HANA connector application URL.

For more information, see Configuration of Business Rules Capability.

- The cloud connector system is connected to your subaccount so you can access the URL (XS UAA and HANA connector application) of your on-premise system.

  For more information, see Cloud Connector.

- The Rules HANA connector is configured.

  For more information, see Configuration on Rules HANA Connector.

Context

You must create two types of destinations on SAP BTP subaccount. The first destination points to the URL of the rule HANA connector and the second destination points to a token URL from which an access token is fetched.

Procedure

1. Log in to the SAP BTP cockpit as administrator.
2. Navigate to the subaccount you want to use for deployment.
3. Navigate to **Connectivity** **Destinations**.
4. Choose **New Destination**.

   For more information, see Create HTTP Destinations.

5. Provide the following information:

Option	Description
Name	<dest_name> **Note** This destination name is used in creating additional property.
Type	HTTP
Description	Description of the destination
URL	Use the XS UAA URL (with protocol as HTTP instead of HTTPS) and append /oauth/token at the end of URL.
Proxy Type	On-premise
Authentication	Basic
User	Use the user client ID.
Password	Use the secret client ID.

Note

To make the resource available, ensure that the cloud connector (**Cloud to On-premise**) is configured as follows.

- **Mapping Virtual to Internal System**

Backend Type	SAP HANA

Protocol	HTTPS
Internal Host	<Host name from the URL>
Internal Port	<Port number from the URL>
Virtual Host	Default
Virtual Port	Default
Principle Type	None
Description	Optional

- Resources Accessible on System
  - URL path: /
  - Access Policy: Path and all subpaths

6. Choose Save.
7. Choose New Destination to create another destination, then provide the following information:

Option	Description
Name	<DEST_NAME>
Type	HTTP
Description	Description of the destination.
URL	Use the application URL (with protocol as HTTP instead of HTTPS).
Proxy Type	On-Premise
Authentication	No Authentication

> **Note**
> Ensure that the cloud connector (Cloud to On-premise) is configured.

8. Create an additional property named **bpm.oauth.token.destination** that uses **<DEST_NAME>** as its value.
9. Choose Save.

Security

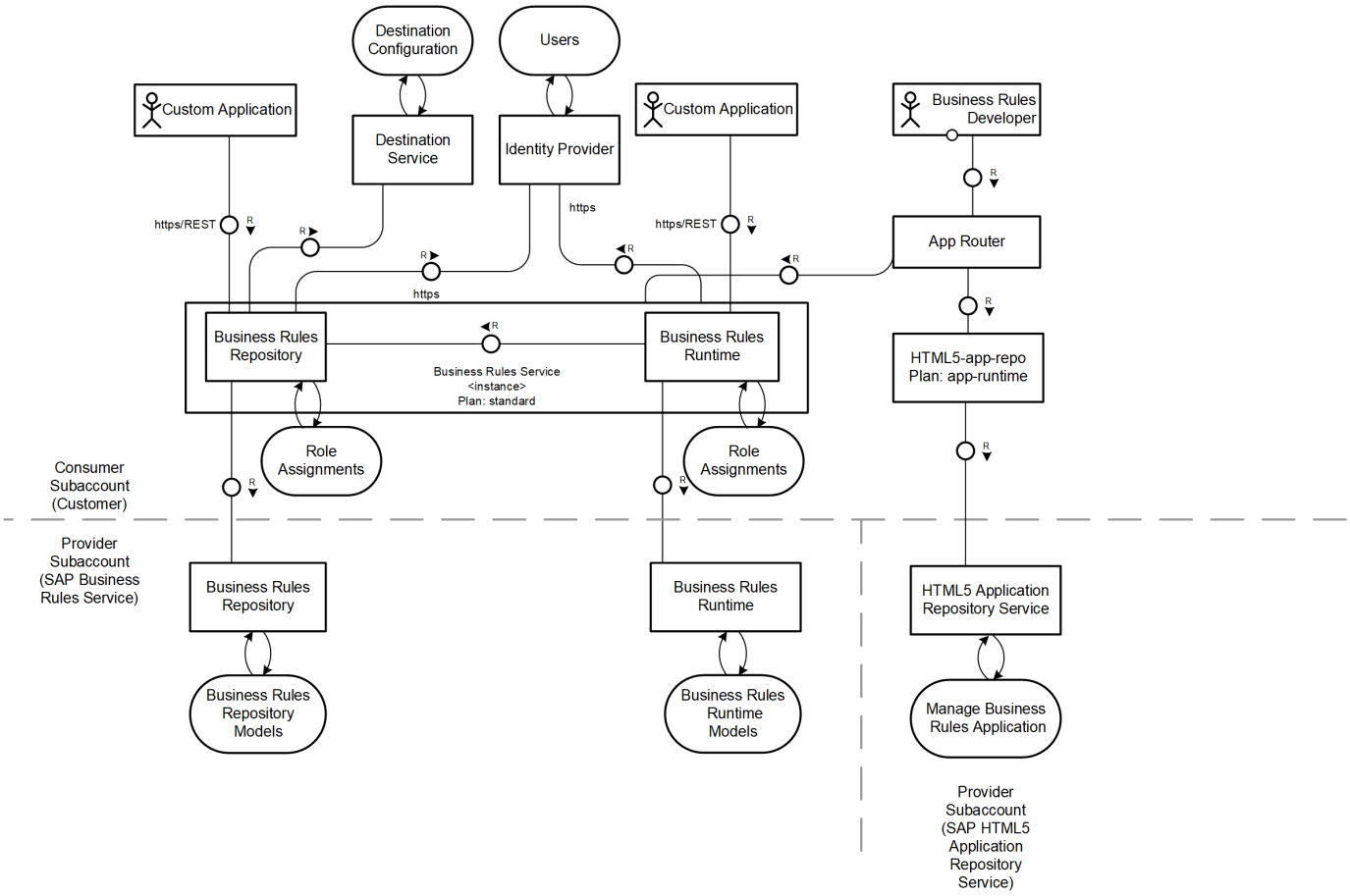
This guide provides an overview of the security-relevant information that applies to the business rules capability.

Related Information

Security Guide

Architecture

The following graphic gives an overview of all components business rules capability.



The service includes the following subservices, which are provisioned into the customer account using the SAP BTP cross-account subscription concept:

- Rules service repository
- Rules service runtime

For more information, see Multitenant Applications.

Prerequisites

- Access to all subservices, requires a valid user identity in the corresponding identity provider configured in the customer account (see Identity Provider and Identity Management).
- Business rules runtime exposes a set of REST-based application programming interfaces (APIs) for modeling rules and invoking the service into the custom application.

  For more information, see Using Business Rules Capability APIs.
- Business Rules Capability is a platform service and it provides a technical role to access the functionality of the service. The application, which consumes the service should ensure access control to the resources created in the Business Rules Capability tenant.

  For granular authorization on business rule resources, the consumer keeps a mapping between custom role and project ID or custom role and rule ID in a resource in their account. For any incoming rule authoring changes, the consumer application checks if the user has custom role. It also checks what resources can this custom role modify in the book keeping. If the incoming request matches with book keeping metadata then the consumers application switches to the technical user with RuleSuperUser role and accesses the service. If the check fails, then the incoming request is rejected at the consumer account. The consuming application can implement the business log functionality to maintain the history of the incoming user authoring the rule changes.

Auditing and Logging Information

Here you can find a list of the security events that are logged by business rules capability.

Security events written in audit logs

Event grouping	What events are logged	How to identify related log events
Security events (`SecurityEventAuditMessage`)	**Rule service deployment** • Rule service deployment triggered • Rule service deployment started • Rule service deployment completed • Rule service deployment failed	**deploy-scheduled** - User <user ID> triggered the deployment of the version <version> of rule service <rule service ID> into the managed system <system ID> in tenant <tenant ID>. **about-to-deploy** - Job for deploying the version <version> of rule service <rule service ID> into the managed system <system ID> in tenant <tenant ID> has been started. **deploy-done** - Job for deploying the version <version> of rule service <rule service ID> into the managed system <system ID> in tenant <tenant ID> has been completed. **deploy-failed** - Job for deploying the version <version> of rule service <rule service ID> into the managed system <system ID> in tenant <tenant ID> has been failed. **For Cloud Runtime:** **about-to-deploy** - Deployment of the rule service <rule service ID> with name <rule service name>, project name <project name> and version <version> of tenant <tenant ID> has been started. **deploy-done** - Deployment of the rule service <rule service ID> with name <rule service name>, project name <project name> and version <version> of tenant <tenant ID> has been completed. **deploy-failed** - Deployment of the rule service <rule service ID> with name <rule service name>, project name <project name> and version <version> of tenant <tenant ID> has been failed.

Event grouping	What events are logged	How to identify related log events
	Rule service undeployment • Rule service undeployment triggered • Rule service undeployment started • Rule service undeployment completed • Rule service undeployment failed	**undeploy-scheduled** - User <user ID> triggered the undeployment of the version <version> of rule service <rule service ID> into the managed system <system ID> in tenant <tenant ID>. **about-to-undeploy** - Job for undeploying the version <version> of rule service <rule service ID> into the managed system <system ID> in tenant <tenant ID> has been started. **undeploy-done** - Job for undeploying the version <version> of rule service <rule service ID> into the managed system <system ID> in tenant <tenant ID> has been completed. **undeploy-failed** - Job for undeploying the version <version> of rule service <rule service ID> into the managed system <system ID> in tenant <tenant ID> has been failed. **For Cloud Runtime:** **about-to-undeploy** - Undeployment of the rule service <rule service ID> with name <rule service name>, project name <project name> and version <version> of tenant <tenant ID> has been started. **undeploy-done** - Undeployment of the rule service <rule service ID> with name <rule service name>, project name <project name> and version <version> of tenant <tenant ID> has been completed. **undeploy-failed** - Undeployment of the rule service <rule service ID> with name <rule service name>, project name <project name> and version <version> of tenant <tenant ID> has been failed.
	Transport rule service • About to transport rule service to managed system • Rule service transport to managed system completed • Rule service transport to managed system failed	**about-to-read** - User <user ID> is about to transport version <version> of rule service <rule service ID> to managed system <system ID> in tenant <tenant ID>. **read-done** - User <user ID> finished transporting version <version> of rule service <rule service ID> to managed system <system ID> of tenant <tenant ID>. **read-failed** - Transport of version <version> of rule service <rule service ID> to managed system <system ID> of tenant <tenant ID> for user <user ID> failed.

Event grouping	What events are logged	How to identify related log events
	Export rule • About to export rule • Rule export completed • Rule export failed	**about-to-read** - User <user ID> is about to export version <version> of rule <rule ID> in tenant <tenant ID>. **read-done** - User <user ID> finished exporting version <version> of rule <rule ID> of tenant <tenant ID>. **read-failed** - Export of version <version> of rule <rule ID> of tenant <tenant ID> for user <user ID> failed.
	Export project • About to export a project • Project export completed • Project export failed	**about-to-read** - User <user ID> is about to export version <version> of project <project ID> in tenant <tenant ID>. **read-done** - User <user ID> finished exporting version <version> of project <project ID> of tenant <tenant ID>. **read-failed** - Export of version <version> of project <project ID> of tenant <tenant ID> for user <user ID> failed.
	Transport project • About to transport a project to a managed system • Project transport to a managed system completed • Project transport to a managed system failed	**about-to-read** - User <user ID> is about to transport version <version ID> of project <project ID> to system <system ID> in tenant <tenant ID>. **read-done** - User <user ID> finished transporting version <version ID> of project <project ID> to system <system ID> of tenant <tenant ID>. **read-failed** - Transport of version <version> of project <project ID> to system <system ID> of tenant <tenant ID> for user <user ID> failed.
	Project purge • Project purge triggered • Project purge started • Projects purge completed • Project purge failed	**purge-scheduled** - User <user ID> triggered the purge of project <project ID> in tenant <tenant ID>. **about-to-purge** - Job for purging of project <project ID> in tenant <tenant ID> has been started. **purge-done** - Job for purging of project <project ID> in tenant <tenant ID> has been completed. **purge-failed** - Job for purging of project <project ID> in tenant <tenant ID> has been failed.

Event grouping	What events are logged	How to identify related log events
	Project deletion • Project deletion triggered • Project deletion started • Project deletion completed • Project deletion failed	**delete-scheduled** - User <user ID> triggered the delete of the version <version ID> of project <project ID> in tenant <tenant ID>. **about-to-delete** - Job for deleting of the version <version ID> of project <project ID> in tenant <tenant ID> has been started. **delete-done** - Job for deleting of the version <version ID> of project <project ID> of tenant <tenant ID> has been completed. **delete-failed** - Job for deleting of the version <version ID> of project <project ID> of tenant <tenant ID> has been failed.
	Data object deletion • Data object deletion triggered • Data object deletion started • Data object deletion completed • Data object deletion failed	**delete-scheduled** - User <user ID> triggered the delete of the version <version ID> of data object <data object ID> in tenant <tenant ID>. **about-to-delete** - Job for deleting of the version <version ID> of data object <data object ID> of tenant <tenant ID> has been started. **delete-done** - Job for deleting of the version <version ID> of data object <data object ID> of tenant <tenant ID> has been completed. **delete-failed** - Job for deleting of the version <version ID> of data object <data object ID> of tenant <tenant ID> has been failed.
	Rule deletion • Rule deletion triggered • Rule deletion started • Rule deletion completed • Rule deletion failed	**delete-scheduled** - User <user ID> triggered the delete of the version <version ID> of rule <rule ID> in tenant <tenant ID>. **about-to-delete** - Job for deleting of the version <version ID> of rule <rule ID> of tenant <tenant ID> has been started. **delete-done** - Job for deleting of the version <version ID> of rule <rule ID> of tenant <tenant ID> has been completed. **delete-failed** - Job for deleting of the version <version ID> of rule <rule ID> of tenant <tenant ID> has been failed.

Event grouping	What events are logged	How to identify related log events
	Rule service deletion • Rule service deletion triggered • Rule service deletion started • Rule service deletion completed • Rule service deletion failed	**delete-scheduled** - User <user ID> triggered the delete of the version <version ID> of rule service <rule service ID> in tenant <tenant ID>. **about-to-delete** - Job for deleting of the version <version ID> of rule service <rule service ID> of tenant <tenant ID> has been started. **delete-done** - Job for deleting of the version <version ID> of rule service <rule service ID> of tenant <tenant ID> has been completed. **delete-failed** - Job for deleting of the version <version ID> of rule service <rule service ID> of tenant <tenant ID> has been failed.
	Ruleset deletion • Ruleset deletion triggered • Ruleset deletion started • Ruleset deletion completed • Ruleset deletion failed	**delete-scheduled** - User <user ID> triggered the delete of the version <version ID> of ruleset <ruleset ID> in tenant <tenant ID>. **about-to-delete** - Job for deleting of the version <version ID> of ruleset <ruleset ID> of tenant <tenant ID> has been started. **delete-done** - Job for deleting of the version <version ID> of ruleset <ruleset ID> of tenant <tenant ID> has been completed. **delete-failed** - Job for deleting of the version <version ID> of ruleset <ruleset ID> of tenant <tenant ID> has been failed.

Related Information

Audit Logging in the Cloud Foundry Environment

Identity Provider and Identity Management

For identity management and authentication, the Business Rules Capability relies on the identity provider (IdP) that is configured in the customer account that owns the respective subscriptions.

All subscriptions requests are authenticated against the identity provider of the customer account. It is authorized against the role assignments specified on the subscriptions in the customer account. Therefore, all users who to interact with the service must be available in the appropriate identity provider. You can replace the default SAP Cloud Identity service with your own corporate identity provider.

For more information about configuring secure access, see Identity and Access Management.

Authorization Configuration

Users must be assigned to a set of roles to use Business Rules Capability.

To assign roles, this service relies on standard functionality of SAP BTP. Accordingly, you assign the roles and permissions to users on the corresponding subscription in the customer account. These assignments are not stored within the provider account.

Roles Required for Runtime and Design-time Services of Business Rules Capability

Role	Description
RulesRepositorySuperUser	This role gives you access to the **Manage Rule Projects** application and repository APIs to model and manage rules.
RulesRuntimeSuperUser	This role gives you access to deploy and execute the rules.

For more information about assigning roles, see Managing Roles.

Data Protection and Privacy

Governments place legal requirements on industry to protect data and privacy. We provide features and functions to help you meet these requirements.

Business rule is a technical object to store the decision logic of an application. It is recommended not to store the personal or sensitive data in the rule expressions. The **Manage Rule Projects** application provides the ID of the user who creates project and modifies it. The user information gets deleted when the project is deleted.

Example

In the following example, the name field contains a dummy value and not the actual name.

Do not hardcode the personal data or personal sensitive data in the rule expressions.

If

Emp_name = "abc" & Emp_gender = "male"

then

Emp_salary = "$500"

Note

SAP does not provide legal advice in any form. SAP software supports data protection compliance by providing security features and data protection-relevant functions, such as blocking and deletion of personal data. In many cases, compliance with applicable data protection and privacy laws is not covered by a product feature. Furthermore, this information should not be taken as advice or a recommendation regarding additional features that would be required in specific IT environments. Decisions related to data protection must be made on a case-by-case basis, taking into consideration the given system landscape and the applicable legal requirements. Definitions and other terms used in this documentation are not taken from a specific legal source

Monitoring and Troubleshooting

Getting Support

Check Platform Status

If you encounter an issue with the service, we recommend that you follow the procedure below

You can follow the availability of the platform at SAP Trust Center. You can check:

To get notifications for updates and downtimes, subscribe at the Cloud System Notification Subscriptions application. Create a subscription by specifying Cloud Product, Cloud Service, and Notification Type. For more information, see Cloud System Notification Subscriptions User Guide.

- the availability by service on the SAP BTP tile of the **Cloud Status** tab page;
- the availability by region on the **Data Center** tab page.

Check Guided Answers

Check the Guided Answers for Business Rules Capability as well as the Guided Answers for SAP BTP.

Contact SAP Support

You can report an incident or error through the SAP Support Portal.

For more information about selected platform incidents, see Root Cause Analyses.

Use the following component for your incident:

Component Name	Component Description
LOD-BPM-RUL	Business Rules Capability

When submitting the incident we recommend including the following information:

- Region information (Canary, EU10, US10)
- Subaccount technical name
- The URL of the page where the incident or error occurs
- The steps or clicks used to replicate the error
- Screenshots, videos, or the code entered

Glossary

This section contains the common terms used in the business rules capability.

Term	Definition
Project	A project is a container that holds the entities of business rules capability. A project contains the following entitites: • Data objects • Rules • Rulesets • Rule Services

Term	Definition
Data Objects	A data object is a reusable entity that represents the data domain of the consuming application. A data object holds the data of your project. Each data object can have many attributes. Attributes add more information to your data object.
Rule	A rule is the technical representation of simple business logic that, once evaluated against live data, leads to a decision. One or more rules can be used to represent an end-to-end business scenario. There are two types of representation of rules: • Decision Table • Text Rule
Rule Service	A rule service is an interface that enables an application to invoke a set of rules associated with it. A rule service also contains the information about the data objects that are used as input and result, the target runtime where the rules are deployed,.
Ruleset	A ruleset links a rule or group of rules to a rule service.
Vocabulary	Vocabulary is a virtual concept that comprises of the basic entities or terms that are used in business rules service.
Expression Language	The language used for modeling a rule expression.
Decision Table	A technical representation of business logic in a tabular format.
Text Rule	A technical representation of business logic in simple if-then format.
Attribute	A set of attributes constitute the data object. It is the data domain within each data object.
Annotation	An annotation defines the context or the target runtime system in which an entity of business rules capability would be used.