Integration Guide | PUBLIC
2026-04-30

Integrating SAP Intelligent Product Recommendation with Other Solutions

Content

1 Introduction to Intelligent Product Recommendations

Integrate your systems with SAP Intelligent Product Recommendation.

You can launch the Intelligent Product Recommendations application from a calling application such as SAP Commerce, or SAP CPQ. It allows you to determine a configurable product through needs selection, and returns the user to the calling application with the selected configuration. You can also use this application for a standalone scenario.

2 Parameters to call Intelligent Product Recommendations

Set these parameters in your applications to call up Intelligent Product Recommendations.

Parameter Name	Description	Example
caller	Name of the calling application	commerce, cpq
userid	Authenticated user of the calling application	
lang	Language used for the callback to the calling application	en
currency	Currency used for the callback to the calling application	USD
base	Base URL for the callback	
path	Base Store	
product	Product category code	SMART

Example

https://<ipr url>/runtimeui/index.html?caller=cpq&userid=&lang=en&currency=USD&base=<cpq base url>&path=<base store path>&product=SMART

3 Integration with SAP Variant Configuration and Pricing

Configuration details to set up SAP Intelligent Product Recommendation integration with SAP Variant Configuration and Pricing.

Destination Configuration

Field	Value
Name (Mandatory)	CPS_DESTINATION
Type (Optional)	HTTP
Description (Optional)	Leave it blank
URL (Optional)	Address as defined in the cloud connector
Proxy Type (Optional)	Internet
Authentication (Optional)	OAuth2ClientCredentials
Client ID (Mandatory)	
Client Secret	
Token Service URL type	
Token Service User	
Token Service Password	

Additional Properties

Set WebIDEEnabled to be true, and select *Use default JDK truststore*.

Related Information

Creating and Configuring a Destination in SAP BTP Cockpit

4 Integrating with SAP CPQ

You can integrate Intelligent Product Recommendations with SAP CPQ using its **External Configurator Integration** capabilities.

Context

This allows you to use the SAP CPQ quoting capabilities while continuing to use your existing configurator. When you configure Intelligent Product Recommendations with SAP CPQ, the SAP CPQ user is given access to both the user interface and capabilities of Intelligent Product Recommendations.

4.1 Setting up Credentials for Intelligent Product Recommendations

You can store credentials for accessing Intelligent Product Recommendations.

Context

When you store a credential entry, you can mention its name in the script that is sent to Intelligent Product Recommendations to perform a basic user authentication.

Procedure

1. Launch the SAP CPQ application.
2. Choose *Setup*.
3. Choose *Security* > *Credential Management*.
4. Choose *New Entry*.
5. In the *New Entry* dialog that opens, enter the following values.

Field Name	Values
Name (mandatory)	A unique value of the credential to distinguish it from others.

Field Name	Values
Type	Credential
Identifier (mandatory)	Client ID of your SAP CPS configuration service.
Secret	Client secret of your SAP CPS configuration service.

6. Choose *Save*.
7. In the *Credential Management* page, switch on the *Active* button.

> ⓘ Note
>
> You can temporarily make a credential entry inactive, preventing its use for authentication in scripts.

Related Information

Credential Management

4.2 Creating Configurable Products

Maintain details of your configurable products in SAP CPQ.

Context

You must have a launching product in the SAP CPQ catalog to call the Intelligent Product Recommendations application.

Procedure

1. Launch the SAP CPQ application.
2. Choose *Setup*.
3. Choose *Product Catalog* > *Products*.
4. Choose *Add New* to create a product.
5. Search for the required product, and choose *Edit*.
6. Enter all the required details in the *Definition* tab.

> ⓘ Note
>
> Enter *Display Type* as *Configurable Product*, and *Configuration Type* as *External*.

7. Choose *Select Category* to assign a category to the configurable product.

> **Note**
>
> Assign product to at least one category for it to appear in the SAP CPQ catalog page. You can either use an existing category or create one, and then assign it to the configurable product.

8. Move on to the *Pricing* tab to define the pricing type that pertains to the product.
9. Move on to the *Attributes* to define the product attributes.

> **Note**
>
> When you select the *Additional Product Attributes* checkbox, the attributes are included in the recommendation, however the configuration will not appear in the SAP CPQ catalog page.

10. Choose *Save*.

    Th system creates the product.

Related Information

Create Products
Categories

4.3 Adding Global Scripts

Add scripts to launch Intelligent Product Recommendations and return to SAP CPQ.

Procedure

1. Launch the SAP CPQ application.
2. Choose *Setup*.
3. In the *Setup Home* screen, choose *Develop* > *Global Scripts*.
4. Choose *Add New* to add a global script for launching Intelligent Product Recommendations.
5. Enter the required details in the *Add New* screen.

Field Name	Description
Name	The name of the script.
System ID	ID of the system that is generated automatically when you enter the script name.

Field Name	Description
Description	Detailed description of the script.
Start Date	The date when you want to start the execution of the script.
End Date	The date when you want to end the execution of the script.
Active	Determines whether or not the script is executed when events occur.

6. In the *Script Code* area, paste the Intelligent Product Recommendations Launch Python script for launching Intelligent Product Recommendations from SAP CPQ.
7. Choose *Save*.
8. Choose *Add New* to add a script for returning to SAP CPQ.
9. Enter the required details in the *Add New* screen.
10. In the *Script Code* area, paste the SAP CPQ Return Python script for returning to SAP CPQ.
11. Choose *Save*.

Related Information

External Configurator Integration for Quote 2.0
Scripting

4.3.1 Intelligent Product Recommendations Launch Script

Python script for launching Intelligent Product Recommendations from SAP CPQ.

> **Note**
>
> The table names are placeholders. As the URL `https://<domain name>.cfapps.eu10-004.hana.ondemand.com/runtimeui/index.html` is hard-corded in the script, only an administrator can modify it. You can also have a fallback URL to call Intelligent Product Recommendations without giving any parameters.

> **Code Syntax**

```
try:
    quoteId = '0'
    selectedMarketCur = 'USD'
    if context.Quote is not None:
        quoteId = str(context.Quote.Id)
        selectedMarketCur = context.Quote.SelectedMarket.CurrencyCode
    quoteItemId = '0'
    if context.EditedItem is not None:
        quoteItemId = str(context.EditedItem.QuoteItem)
    iprSetting = SqlHelper.GetSingle('select top 1 * from
<iprRedirectionURLConfigTable>')
```

```
    iprProduct = SqlHelper.GetList("select * from
<cpqProductToIprProductCategoryMappingTable> where product = '" +
str(context.Product.PartNumber) + "'")
    if iprSetting is not None:
        if iprProduct[0].category is not None:
            redirectionUrl = iprSetting.url + '?p1=' + iprSetting.p1 +
'&p2=&p3=en/' + selectedMarketCur + '&p4=' + iprSetting.p4 + '&p5=' +
iprSetting.p5 + '&p6=' + iprProduct[0].category + '&p7=' + quoteId
        elif iprSetting.p6 is not None:
            redirectionUrl = iprSetting.url + '?p1=' + iprSetting.p1 +
'&p2=&p3=en/' + selectedMarketCur + '&p4=' + iprSetting.p4 + '&p5=' +
iprSetting.p5 + '&p6=' + iprSetting.p6 + '&p7=' + quoteId
        else:
            redirectionUrl = iprSetting.url + '?p1=' + iprSetting.p1 +
'&p2=&p3=en/' + selectedMarketCur + '&p4=' + iprSetting.p4 + '&p5=' +
iprSetting.p5 + '&p7=' + quoteId
    else:
        redirectionUrl = 'https://<domain name>/runtimeui/index.html'
except Exception as ex:
    Log.Error('Error in script for configuring iQuote product: {}'.format(ex))
```

4.3.2 SAP CPQ Return Script

Python script to return to the SAP CPQ application from Intelligent Product Recommendations.

> **Note**
>
> The table name is a placeholder.

Code Syntax

```
def GetExternalParameter(externalParameters, parameterName):
    if parameterName not in externalParameters:
        Trace.Write('{} is not in script parameters.'.format(parameterName))
        return None
    else:
        return externalParameters[parameterName]
# returns quote or creates new one if one does not exist
def GetQuote(quoteHelper, quoteId):
    try:
        Trace.Write("Trying to get quote with id: {0}".format(quoteId))
        quote = quoteHelper.Get(quoteId)
        Trace.Write("Quote found. QuoteId: {0}".format(quoteId))
        return quote
    except:
        quote = quoteHelper.CreateNewQuote()
        Trace.Write("New quote created. QuoteId: {0}".format(quote.Id))
        return quote
# gets valid access token from the custom table. If token expiration date
passed returns None
def GetAccessTokenFromStore():
    accessToken = None
    storedToken = SqlHelper.GetSingle('select top 1 * from
<iprToCpsAuthTokenStorageTable>')
    if storedToken is not None and storedToken.ExpirationTime >
DateTime.UtcNow:
        accessToken = storedToken.Token
        Trace.Write('Access token from store: {}'.format(accessToken))
    return accessToken
# saves access token and token expuiration date to the custom table
```

```
def StoreAccessToken(token, expiresIn):
    expirationTime = str(DateTime.UtcNow.AddSeconds(expiresIn))
    storedTokenId = 0
    storedToken = SqlHelper.GetSingle('select top 1 CpqTableEntryId from
ipr_auth')
    if storedToken is not None:
        storedTokenId = storedToken.CpqTableEntryId
    table = SqlHelper.GetTable('ipr_auth')
    table.AddRow({ "CpqTableEntryId" : storedTokenId, "Token" : str(token),
"ExpirationTime" : expirationTime })
    SqlHelper.Upsert(table)
    Trace.Write('New access token stored. Expiration time: {}, Token:
{}'.format(expirationTime, token))
# gets auth token from IPR and caches it
def GetNewAccessToken():
    accessToken = ''
    authUrl = '{}?grant_type={}'.format(AUTH_URL, AUTH_PARAM_GRANT_TYPE)
    response = AuthorizedRestClient.Get(CREDENTIAL_STORE, authUrl)
    if response is not None:
        accessToken = response.access_token
        StoreAccessToken(accessToken, response.expires_in)
        Trace.Write('Access token: {}'.format(accessToken))
    return accessToken
# gets the auth token for authorizing on CPS configuration service
def GetAccessToken():
    accessToken = GetAccessTokenFromStore()
    if accessToken is None:
        accessToken = GetNewAccessToken()
    return accessToken
# creates and returns auth headers in a form of dictionary to be passed to
RestClient
def CreateAuthorizationHeaders(accessToken):
    headers = {}
    headers["Authorization"] = "Bearer {}".format(accessToken)
    sapPassport = SapPassport.GeneratePassport('Create Configuration')
    headers["sap-passport"] = sapPassport.Value
    Trace.Write('Authorization headers created: {}'.format(str(headers)))
    return headers
# authenticates and returns external configuration
def GetConfiguration(configurationId):
    accessToken = GetAccessToken()
    headers = CreateAuthorizationHeaders(accessToken)
    #headers['Cookie'] = "lbhash=\"{}\"".format(lbhash)
    #requestUrl = "{}/{}/external?
ShortTextandItemId.isRequired=true".format(CONFIG_URL, configurationId)
    requestUrl = "{}/{}/".format(CONFIG_URL, configurationId)
    response = RestClient.Get(requestUrl, headers)
    Trace.Write("Configuration retrived for request URL {} ->
{}".format(requestUrl, str(response)))
    return response
# CPQ catalog URL
CPQ_CATALOG_URL = 'https://<CPQ domain name>/Catalogue/CategoryTree.aspx'
# CPQ quote URL
CPQ_QUOTE_URL = 'https://<CPQ domain name>/cart/edit?cartcompositenumber={}'
# authentication URL for CPS server
AUTH_URL = 'https://<CPS domain name>.authentication.eu10.hana.ondemand.com/
oauth/token'
# authentication parameters
AUTH_PARAM_GRANT_TYPE = 'client_credentials'
AUTH_PARAM_SCOPE = 'uaa.resource'
CREDENTIAL_STORE = '<Name of your credential that you create in Credential
Management>'
# configuration URL to CPS server
CONFIG_URL = 'https://<CPS server>.cfapps.eu10.hana.ondemand.com/api/v2/
configurations'
try:
    externalParameters = context.ExternalParameters
    # get product related information
```

```
    configId = GetExternalParameter(externalParameters, 'configId')
    variantId = GetExternalParameter(externalParameters, 'MATERIALVARIANT')
    # Trace.Write("Config Id: " + configId)
    # Trace.Write("Material Variant Id: " + variantId)

    # get quote related information
    # Trace.Write("Get Quote Nr ")
    quoteId = GetExternalParameter(externalParameters, 'salesdocnumber')
    Trace.Write("Quote Id: " + quoteId)

    # get quote
    quote = GetQuote(QuoteHelper, int(quoteId))

    if configId is not None and configId != '0':
        # get configuration
        cpsJson = GetConfiguration(configId)
        # extract configXML and then remove its content in original payload
        externalConfiguration = str(cpsJson)
        # get product
        product = cpsJson.rootItem.key
        productId = str(product)
        Trace.Write("Product Id: " + productId)
        inlineProduct = ProductHelper.CreateProduct(productId)
        counter = 0
        while counter < cpsJson.rootItem.characteristics.Count:
            i = 0
            if str(cpsJson.rootItem.characteristics[counter].id) != 'VARKOND':
                if cpsJson.rootItem.characteristics[counter].values.Count > 0:
                    cstic = str(cpsJson.rootItem.characteristics[counter].id)
                    csticVal =
str(cpsJson.rootItem.characteristics[counter].values[0].value)
                    Trace.Write("Cstic: " + cstic + " Value: " + csticVal)

inlineProduct.Attributes.GetBySystemId(cstic).SelectValue(csticVal)
            counter += 1
        # add product to quote
        Trace.Write("Start to add Item ")
        quote.AddItem(inlineProduct, 1)
        Trace.Write('Product added to the quote')

        redirectionUrl = CPQ_QUOTE_URL.format(quote.QuoteNumber)
    elif variantId is not None and variantId != '0':
        productId = str(variantId)
        Trace.Write("Variant Id: " + productId)
        inlineProduct = ProductHelper.CreateProduct(productId)

        # add product to quote
        Trace.Write("Start to add Item ")
        quote.AddItem(inlineProduct, 1)
        Trace.Write('Product added to the quote')
        redirectionUrl = CPQ_QUOTE_URL.format(quote.QuoteNumber)
    else:
        redirectionUrl = CPQ_CATALOG_URL
except Exception as ex:
    Trace.Write("Error on landing to CPQ: {}".format(ex))
    Log.Error("Error on landing to CPQ: {}".format(ex))
```

You can find the authentication URL for CPS server and configuration URL by choosing *Setup* and *Providers* > *SAP Integrations*. Choose *SAP Variant Configuration and Pricing*. In the *Authentication Settings*, you can view the URL in the *Authentication URL*. For configuration URL, choose *Base URLs*, and you can view the URL in the *Configuration Base URL* field.

4.4 Attaching SAP IPR Scripts for third party Integration in SAP CPQ

You can use the external configurator to configure SAP Intelligent Product Recommendation in SAP CPQ.

Prerequisites

You've ensured that you are on the Quote 2.0 engine.

Procedure

1. Launch the SAP CPQ application.
2. In the *SAP CPQ Setup Home* screen, choose *Providers* > *Third-Party Integrations*.
3. Choose *External Configurator*.
4. In the *External Configurator Provider Settings* screen, perform the following substeps:
   a. In the *Land to SAP CPQ* section, select the **SAP CPQ Return Script** in the *Global Script* dropdown list, and choose *Attach* and *Save*.
   b. In the *Configure* section, select the Intelligent Product Recommendations Launch Script in the *Global Script* dropdown list and choose *Attach* and *Save*.

4.5 Creating Custom Tables for Intelligent Product Recommendations in SAP CPQ

You can use data from nonstandand SAP CPQ tables during the configuration process by creating your own custom tables.

Context

Custom tables let you maintain your own data that you want to use for configuration purpose.

> ⓘ Note
>
> You must create three custom tables to maintain Intelligent Product Recommendations information in SAP CPQ. The tables are:
>
> - One for your configurable product.

- One for your domain information.
- One for authorization.

Creating Custom Table for Intelligent Product Recommendations [page 14]
Maintain values for Intelligent Product Recommendations in its own custom table.

Creating Custom Table for Intelligent Product Recommendations Domain [page 15]
Maintain Intelligent Product Recommendations specific integration details in its own custom table.

Creating Custom Table for Intelligent Product Recommendations Authorization [page 16]
Maintain Intelligent Product Recommendations authorization details in its own custom table.

Related Information

Custom Tables

4.5.1 Creating Custom Table for Intelligent Product Recommendations

Maintain values for Intelligent Product Recommendations in its own custom table.

Context

Use this table to create the mapping between the launching product in SAP CPQ and the product category of Intelligent Product Recommendations.

Procedure

1. Launch the SAP CPQ application.
2. Choose *Setup*.
3. Choose *Product Catalog* > *Custom Tables*.
4. Choose *Define New* to create your own custom table.
5. Enter a name of the table that you want to create in the *Name* field, and select *No Audit* from the *Audit Trail level* dropdown list.
6. Enter the following information for your table.

For Intelligent Product Recommendations

Column Name	Column Type	Length/Decimal Places	Is Nullable	Personally Identifiable Information	Suppress Information Logging
product	NVARCHAR	250	Yes	-	-
category	NVARCHAR	250	Yes	-	-

7. Choose *Save*.
8. Once the table is created, search for it in the *Custom Tables* page.
9. To add values to your table, choose *View entries*.
10. In the *View Table Entries:* `<table name>` page, choose *Add New*.
11. Enter values for the columns that you've configured for this table.
12. Choose *Save*, or *Save and New* to continue entering values for your table.

4.5.2 Creating Custom Table for Intelligent Product Recommendations Domain

Maintain Intelligent Product Recommendations specific integration details in its own custom table.

Procedure

1. Launch the SAP CPQ application.
2. Choose *Setup*.
3. Choose *Product Catalog* > *Custom Tables*.
4. Choose *Define New* to create your own custom table.
5. Enter a name of the table that you want to create in the *Name* field, and select *No Audit* from the *Audit Trail level* dropdown list.
6. Enter the following information.

   For Intelligent Product Recommendations Domain

Column Name	Column Type	Length/Decimal Places	Is Nullable	Personally Identifiable Information	Suppress Information Logging
url	NVARCHAR	250	Yes	-	-
p1	NVARCHAR	250	Yes	-	-
p4	NVARCHAR	250	Yes	-	-
p5	NVARCHAR	250	Yes	-	-
p6	NVARCHAR	250	Yes	-	-

7. Once the table is created, search for it in the *Custom Tables* page.

8. To add values to your table, choose *View entries*.
9. In the *View Table Entries:* `<table name>` page, choose *Add New*.
10. Enter values for the columns that you've configured for this table.

    Table Values

Field Name	Values
url	URL of the integrated Intelligent Product Recommendations application.
p1	**cpq**
p4	Your domain URL
p5	integration/external-configurator/external/landing
p6	

11. Choose *Save*, or *Save and New* to continue entering values for your table.

4.5.3 Creating Custom Table for Intelligent Product Recommendations Authorization

Maintain Intelligent Product Recommendations authorization details in its own custom table.

Context

This table stores the access token for CPS, which is requested using the SAP CPQ Return Script. When you run the script, it searches for a timely valid access token in this table. If such a token is not available, then the script requests a new access token from authorization URL, and stores it in this table including the token's expiration time.

Procedure

1. Launch the SAP CPQ application.
2. Choose *Setup*.
3. Choose *Product Catalog* > *Custom Tables*.
4. Choose *Define New* to create your own custom table.
5. Enter a name of the table that you want to create in the *Name* field, and select *No Audit* from the *Audit Trail level* dropdown list.
6. Enter the following information.

For Intelligent Product Recommendations Authorization

Column Name	Column Type	Length/Decimal Places	Is Nullable	Personally Identifiable Information	Suppress Information Logging
Token	NVARCHAR	2500	Yes	-	-
ExpirationTime	DATETIME		Yes	-	-

7. Choose *Save*.
8. Once the table is created, search for it in the *Custom Tables* page.
9. To add values to your table, choose *View entries*.
10. In the *View Table Entries:* `<table name>` page, choose *Add New*.
11. Enter values for the columns that you've configured for this table.
12. Choose *Save*, or *Save and New* to continue entering values for your table.

5 Integrating with SAP Sales Cloud

You can integrate Intelligent Product Recommendations with SAP Sales Cloud using its **Mashup Authoring** capabilities.

Context

This allows you to use the SAP Sales Cloud opportunity and quoting capabilities. When you configure Intelligent Product Recommendations with SAP Sales Cloud, the SAP Sales Cloud user is given access to both the user interface and capabilities of Intelligent Product Recommendations.

5.1 Prerequisites to Integrate Intelligent Product Recommendations with SAP Sales Cloud

List of prerequisites that need to be in place before you can integrate Intelligent Product Recommendations and and SAP Sales Cloud.

- Ensure that document type **ZQ** for creating budgetary quote is maintained in SAP Sales Cloud.
- Ensure that the currency of the product data in Intelligent Product Recommendations and SAP Sales Cloud is same.
- Ensure that Intelligent Product Recommendations domains are whitelisted in SAP Sales Cloud, and SAP Sales Cloud domains are whitelisted in your SAP BTP Cockpit subaccount.

> ⓘ Note
>
> - To add Intelligent Product Recommendations domains for whitelisting, go to *User Menu* > *Settings* > *All Settings* and under *General*, choose *Content Security Policy*, and add your domains in the *Script Source*, *Style Source*, *Default Source*, *Frame Source*, and *Image Source* sections.
> - To add SAP Sales Cloud domains for whitelisting, go to your subaccount in SAP BTP Cockpit, and choose *Security* > *Settings*. In the *Trusted Domians* tab, add your SAP Sales Cloud domain by choosing *Add* in the *Domain* section.

5.2 Authoring Mashup for Intelligent Product Recommendations

Use SAP Sales Cloud's mashup capabilities to integrate with Intelligent Product Recommendations.

Procedure

1. Launch SAP Sales Cloud.
2. Go to *User Menu* > *Settings* > *All Settings* and under *Extensibility*, choose *Mashup Authoring*.
3. In the *Create Mashup* screen area, enter a value in *Mashup Name* (mandatory) and select *Status* as **Active**.
4. In the *Input Parameters* screen area, provide the URL of Intelligent Product Recommendations so that you want to integrate it with SAP Sales Cloud.
5. In the *Parameter, Constant and API Key* section, choose *+ Add Parameter* to add a paramter, and provide the following values.

Field	Value
First field	p1
Second field	SALESCLOUD

6. Add another parameter by choosing +, and provide the following values.

Field	Value
First field	p7
Second field	Leave it empty

7. Choose *Save*.

5.3 Adding Intelligent Product Recommendations as a Tab

Add Intelligent Product Recommendations as a tab in SAP Sales Cloud.

Prerequisites

Ensure that you have the ID of an Intelligent Product Recommendations opportunity.

Procedure

1. Launch SAP Sales Cloud, and search for an Intelligent Product Recommendations opportunity.
2. Go to *User Menu* > *Start Adaptation*.
3. Hover over the area where you see an orange + *add icon*, and choose *edit icon*.

   The system display the list of available mashups.
4. Choose > *navigation right arrow* of the Intelligent Product Recommendations mashup that you want to add as a tab, and choose *Apply*.

   The system displays the embedded Intelligent Product Recommendations app.
5. To popuplate data in the Intelligent Product Recommendations tab, perform the following steps.
   a. Hover over the area where you see an orange + *add icon*, and choose *edit icon*.
   b. In the *Edit Properties* dialog, select **General/Technical ID** as the value for the *p7* field, and choose *Apply*.

      > ⓘ Note
      >
      > The technical ID is the Intelligent Product Recommendations opportunity ID.

   c. Choose *End Session*.

Important Disclaimers and Legal Information

Hyperlinks

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

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

Videos Hosted on External Platforms

Some videos may point to third-party video hosting platforms. SAP cannot guarantee the future availability of videos stored on these platforms. Furthermore, any advertisements or other content hosted on these platforms (for example, suggested videos or by navigating to other videos hosted on the same site), are not within the control or responsibility of SAP.

Beta and Other Experimental Features

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

Example Code

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

Bias-Free Language

SAP supports a culture of diversity and inclusion. Whenever possible, we use unbiased language in our documentation to refer to people of all cultures, ethnicities, genders, and abilities.

www.sap.com/contactsap