# CPI Development Standards

|  |
| --- |
| ***SR Technics*** |
|  |
| ***CPI Development Standards*** |
| ***Version 1.0*** |
|  |
| Date: 19th December 2022 |

**Document Control**

|  |  |  |  |
| --- | --- | --- | --- |
| ***Owner:*** | HCL | ***Status:*** | For Approval |

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| ***Version*** | ***Description of Change*** | ***Date*** | ***Author*** |
| 1.0 | Initial draft for layout and core content |  | Ibrahim Shaik/Paul Bartlett |

**Reviewers**

|  |  |
| --- | --- |
| ***Name*** | ***Position/Role*** |
| Paul Bartlett | Digital Platform Lead (HCL) |
| Patrick Lau | Digital Platform Lead (SRT) |

**Distribution List – For Information Only**

|  |  |
| --- | --- |
| ***Name*** | ***Position/Role*** |
|  |  |
|  |  |
|  |  |

**Table of Contents**

[**1** **Introduction**](#CPIDevelopmentStandards-_Toc122390804)   
[1.1 SR Technics Background](#CPIDevelopmentStandards-_Toc122390805)   
[1.2 Purpose of this document](#CPIDevelopmentStandards-_Toc122390806)   
[1.3 Scope](#CPIDevelopmentStandards-_Toc122390807)   
[1.4 Target Audience](#CPIDevelopmentStandards-_Toc122390808)   
[1.5 References](#CPIDevelopmentStandards-_Toc122390809)   
[1.6 Glossary](#CPIDevelopmentStandards-_Toc122390810)   
[1.7 Links](#CPIDevelopmentStandards-_Toc122390811)   
[**2** **Development Standards**](#CPIDevelopmentStandards-_Toc122390812)   
[2.1 General Principles](#CPIDevelopmentStandards-_Toc122390813)   
[2.2 Common Principles](#CPIDevelopmentStandards-_Toc122390814)   
[2.3 Third Party Integration Principles](#CPIDevelopmentStandards-_Toc122390815)   
[2.4 Architecture & Design](#CPIDevelopmentStandards-_Toc122390816)   
[2.4.1 Development Tool](#CPIDevelopmentStandards-_Toc122390817)   
[2.4.2 Support (Operation Model)](#CPIDevelopmentStandards-_Toc122390818)  
[2.5 Environment](#CPIDevelopmentStandards-_Toc122390820)   
[2.5.1 Availability](#CPIDevelopmentStandards-_Toc122390821)   
[2.6 Monitoring, alerting and error handling](#CPIDevelopmentStandards-_Toc122390822)   
[2.7 Common Components & Re-use](#CPIDevelopmentStandards-_Toc122390823)   
[2.8 Integration Patterns](#CPIDevelopmentStandards-_Toc122390824)   
[2.9 Naming Conventions](#CPIDevelopmentStandards-_Toc122390825)   
[2.10 Coding](#CPIDevelopmentStandards-_Toc122390826)   
[2.11 Script Guidelines](#CPIDevelopmentStandards-_Toc122390827)   
[2.12 Source Control](#CPIDevelopmentStandards-_Toc122390828)   
[2.13 Unit / Link Testing](#CPIDevelopmentStandards-_Toc122390829)   
[2.14 Testing with S4 environment:](#CPIDevelopmentStandards-_Toc122390830)   
[2.15 Quality assurance & Review](#CPIDevelopmentStandards-_Toc122390831)   
[2.16 Security](#CPIDevelopmentStandards-_Toc122390833)   
[**3** **Appendices**](#CPIDevelopmentStandards-_Toc122390834)   
[3.1 Good practices and guidance](#CPIDevelopmentStandards-_Toc122390835)   
[3.2 Example solutions / templates](#CPIDevelopmentStandards-_Toc122390836)   
[3.3 Content design guidance](#CPIDevelopmentStandards-_Toc122390837)

# Introduction

## SR Technics Background

The management of SR Technics has decided to initiate a strategic initiative – program ReSeT - to fundamentally reshape and digitally transform the way SR Technics conducts its operations in the new organizational setup focused on engines maintenance only.

## Purpose of this document

The purpose of this document is to deliver a consistent set of development standards for producing and supporting SAP Business Technology Platform Integration Suite (BTP-IS) integration solutions and/or wider SAP Integration Suite solutions. Going forward within the document the term 'BTP-IS' will be used to cover the wider range of these SAP integration solutions.

* SRT Technics has a strong preference for Fit to Standard S/4 iMRO/4 so SAP Standard Integration.​
* SAP Integration Suite provides Standard SAP to SAP and SAP to 3rd Party Application Integration​
* SRT ReSeT seeks to minimize non-Standard Integration enhancements​

This document will be used to ensure all integration developments adhered to common core set of policies, standards and best practice guidance. These standards include general integration standards and also those specific to SAP's BTP-IS.  
The aim of the standards is to help drive a consistent design, style and structure to Integration developments and to ensure they are implemented within BTP-IS component and solutions as they are both built and maintained.   
It should be noted that BTP-IS developments utilize the development tools and practices inherent within the BTP-IS tool kit. As such, developments and integration flows are built up using graphical elements to depict the process flow and also mapping can also be via graphical tools, as such the direct coding is more limited to 'code fragments' or specialized functions called in processing or mapping.

## Scope

The scope of this document is to include:

* Guidelines on design, style and structure of BTP-IS developments
* Consistent set of configuration/programming guidelines
* Consistent naming conventions for development objects.
* Rules based on SAP & Industry best practices around integration
* Rules for writing code that is; efficient, simple to support and when applicable re-usable.
* Configuration/Development hints and information based on best practices
* Guidelines when interacting with third party integration layers

## Target Audience

It is essential that every person involved in the development process familiarizes him/herself with this document and applies the described standards and processes.   
Key target audiences are

* Integration Architects working with BTP-IS, either for design work, governance/approval or development.
* BTP-IS Analysts and Developers
* Integration managers delivering BTP-IS solutions
* Awareness for functional/technical resources working in the integration solutions

## References

|  |  |  |  |
| --- | --- | --- | --- |
| ***Ref*** | ***Description of Document*** | ***Doc Name*** | ***Version*** |
| [Ref 1] | Integration Strategy |  |  |
| [Ref 2] | Application Architecture / System Landscape |  |  |
| [Ref 3] | Design Patterns | Integration Strategy |  |
| [Ref 4] | SAP's Dev Guide for BTP-IS | <https://help.sap.com/doc/dd250f2e3c2645a8ae327e935071281e/latest/en-US/DevGuide_ManageIntContent_External.pdf> |  |

## Glossary

|  |  |
| --- | --- |
| ***Term*** | ***Description*** |
| ABAP | SAP's 4GL Language - Advanced Business Application Programming |
| API | Application Programming Interface |
| APIM | API Management |
| BAPI | Business API (an SAP API method/framework) |
| BAS | Business Application Studio |
| BTP | Business Technology Platform |
| CAP | Cloud Application Programming (Open source), see RAP |
| CDS | Core Data Service View |
| DA | Design Authority |
| FT | Finance Transformation Project with SRT |
| IDE | Integrated Development Environment |
| IDOC | Internal Document (SAP message structure/method) |
| iFlow | Integration Flow |
| IS | Integration Suite |
| ODATA | Rest Based Open Data Protocol |
| PSE | Personal Security Environment |
| RAP | Restful Application Programming (preferred for Abap in cloud), see also CAP |
| RFC | Remote Function Call |
| S/4HANA | SAP HANA based ERP platform |
| SOAP | Simple Object Access Protocol |

## Links

|  |  |  |
| --- | --- | --- |
| ***Ref*** | ***Description*** | ***Link*** |
| Link1 | SAP API Business Hub | <https://api.sap.com/> |
| Link2 | SAP Best Practices | <https://rapid.sap.com/bp/> |
| Link3 | CIO Guide (on integration) | <https://www.sap.com/documents/2018/06/de3238a0-077d-0010-87a3-c30de2ffd8ff.html> |
| Link4 | SAP Security Guide | [https://help.sap.com/docs/SAP\_INTEGRATION\_SUITE/51ab953548be4459bfe8539ecaeee98d/a58b2400b3094009988a53b0a63b455a.html](https://help.sap.com/docs/SAP_INTEGRATION_SUITE/51ab953548be4459bfe8539ecaeee98d/a58b2400b3094009988a53b0a63b455a.html+) |
| Link5 | SAP JavaScript coding guidelines | [https://help.sap.com/saphelp\_snc700\_ehp04/helpdata/en/ed/ed636b85584cd586b1fe231d2b5dac/content.htm?no\_cache=true](https://help.sap.com/saphelp_snc700_ehp04/helpdata/en/ed/ed636b85584cd586b1fe231d2b5dac/content.htm?no_cache=true+) |
| Link6 | Groovy coding guidelines | <http://groovy-lang.org/style-guide.html> |
| Link7 | Jira tool for SAP Documentation and Test Scripts |  |

# Development Standards

## General Principles

These general principles will be used as guide along with the following key reference documents while designing integration solution in BTP-IS to align with the wider project integration strategy. Therefore, key associated documents are:

* Integration Approach/Strategy [Ref1]
* High Level Architecture Document [Ref2]

Additionally, there are various public reference guides supplied by SAP and these also provide further technical reference details for integration, these are:

* SAP API Business Hub [Link1]
* SAP Best Practices [Link2]
* CIO guide to assist in developing integration solution [Link3]

## Common Principles

The following are common principles applied to all BTP-IS developments and components:

1. General Middleware principles should be applied as per the Integration Approach/Strategy [Ref1], High Level Architecture [Ref2] to align with the Projects integration strategy.
2. Components must be built for reuse whenever possible (including elements, artefacts, iFlows and template etc.) so that these can be used across other integration flows. Example security artefacts, function library etc. Any components that are expected to be used for wider re-use (i.e., common error or exception handling) then these need to be fully documented on the generic functions and usage.
3. All BTP-IS developments need to have a functional and technical design with formal test cases. The name of these documents, formats and contents may vary by project but will need pre-agreement.
4. Standard protocol-based convertors/adapters are expected to be used wherever possible (i.e., HTTPs, SFTP etc.) Proprietary connectors/adapters can be used if licensed and available to the Project (such as to SAP via RFC).
5. The iFlow design will be environment neutral and only the actual connection configuration/mapping will have environment and system specific values.
6. Integration Flows (iFlows) will be designed working from the left to right, with usually a single source (on the left) and single target on the right, although multiple systems can be utilised but usually grouped on left for input, right for output and below if used in reference or interim state processing.
7. Data must not be held in the BTP-IS layer (service in SCP).
8. Processing logic in BTP-IS can be broken down into multiple sub processes for better visibility. Use local integration process for any sub process with suitable descriptions.
9. Use suitable and correct format comments in lower-level code (groovy / java script), see Link5 and Link6 for further details. Inline comments are discouraged in SRT. Most use case of inline comments are removed by refactoring to the right level of granularity and self-explanatory names.
10. Consideration to use Standard integration content (editable standard content), Extending the standard iFlows or custom iFlows should be discussed/reviewed by lead if not DA on a case-by-case basis and these items must have DA approval (either at the Functional Specification stage or Technical Specification stage).

## Third Party Integration Principles

* Refer the Integration Approach/Strategy [Ref1] and High-Level Architecture document [Ref2] for decision matrix on selecting an integration approach with a third party.
* A forward-looking preference is towards API/web service driven integration approach and more event-based integrations. Revert to file-based integration approach upon agreement/approval from DA (in case integration approach is not specified in functional specification). Equally current integration will only be changed as required or when clear benefits can be detailed, and the project (or support) workload can cover the re-work/re-design.

## Architecture & Design

The architecture of any solution must be approved by SRT and Project Solutions Architect(s), this may occur for specific developments/changes or be agreed for an overall Project/Programme covering a range of developments.  
The design must also follow agreed Integration Patterns and agreed SAP BTP-IS design templates/standards documented within this document. Any exceptions to these practices would need prior approval via a Solution Architect and/or relevant Design Authority.

### Development Tool

SAP is recommending to use the WEB IDE, and will discontinue support for Eclipse in the near future, so the BTP-IS team will utilise WEB IDE for development.

### Support (Operation Model)

The SRT Operation Model for support needs to be considered while developing solution in BTP-IS to minimize the time required to track, resolve production errors post go live.

* Interface Design should consider incorporating notification and logging to have better operational support as per requirement stated under functional specification document.
* Cloud ALM will be used for monitoring of all SAP environments in SRT landscape. Standard solution is available to integrate BTP-IS for monitoring, auditing, and alerting.
* Custom solution can be built in BTP-IS to expose monitoring logs for better visibility and monitoring of entire business process.

## Environment

The available SAP BTP-IS environments are:

* Development - This is the actual Development /SIT environment
* QA – QA/UAT Environment
* Production – Live environment.

### Availability

Interface design should consider the availability factors and incorporate exception handling as well.  
High Availability (HA) configuration for the Production system will deliver resilience and is covered as per the agreed scope and Technical Architecture that is established with the BTP-IS production environment see [Ref5].

## Monitoring, alerting and error handling

Monitoring strategy is as Integration Strategy.   
Logging and Error Handling:

* Monitoring logs are collected via standard solution.
* Local exception integration process should be built into integration solution which will be aligned with functional requirement in Functional specification and middleware design document.
* Exceptions/errors are captured in standard exception and log routines and shown in BTP-IS monitoring or logs. Additional logging, notification (email) or error message logging can be built as services, but standard BTP-IS functionality is currently the preferred method. Additional log analysis (in Splunk or equivalent is possible and needs discussion)  
    
  Alerting:
* BTP-IS will be configured to raise alerts for any errors. It will alert operation team for any issues.
* Emails will be sent from BTP-IS to operation team with details on error occurred during processing, validation as per business requirement in FS document. BTP-IS will be used to send alert emails in case of general errors and connectivity failure. Email should not be sent out from each process. Email should only be sent out from the event monitoring tool as per the configured policy.
* Validation/mandatory checks will be performed in BTP-IS and exceptions will be raised to align with Functional design document.

Tracing:

* Interface design should consider including sufficient information to be available in trace to enable efficient debugging during incidents.
* Different tracing levels are supported by BTP-IS system – debug, warning, info and error etc.
* Default trace level for production environment is "info".
* We can request for debug trace level to be activated for production by raising request to SAP. It will be done for 1 day
* All trace levels are supported for non-production environment, the default will also be that whilst a higher level can be set in Dev this will have short initial period (i.e., 30 min) to enable a task to be run and analyzed but defaulting back to 'info' after the relevant time period.

Monitoring:

* BTP IS monitoring is the standard solution for monitoring of interfaces which can be integrated with BTP-IS via standard solution.
* BTP-IS logs will be monitored via WEB UI.

Persistence:

* Monitoring logs are stored by default for 30 days which can be increased for persistence upon request to SAP to maximum of 90 days.
* Monitoring logs can be temporarily stored into Data store operations for access within integration flows.

## Common Components & Re-use

The strategy of common components for integration solution and BTP-IS includes but is not limited to:

* Solution will be designed to expose/identify common API(s) to be used across different systems and business scenarios.
* Security artifacts should be maintained under BTP-IS centralized security storage which will be re-used across different integration flows/solutions. (Basis will be the only team that can access and update these in the productive instance, additionally, Basis will also be expected to complete this in the non-production environment as well)
* Scripts should be written/maintained to be re-used across all integration flows if applicable.
* BTP-IS elements – validations, templates are created for re-use
* Communication channels can be created to be re-used with same systems if applicable.
* Custom code can be maintained in common function library to be re-used across all development objects.
* Value mapping artifacts can be created once to be used across all integration flows.
* Message and XSLT mappings can be used across integration flows.
* XSD's and WSDL files can be used across integration flows.

## Integration Patterns

Integration patterns are covered under Integration strategy document [Ref1] and Architecture section.

## Naming Conventions

Below are the details given on the Naming convention principles which need to be followed while developing an object. It is SAP and SAP BTP-IS best practice to have naming standards for all component parts and the standards below adopt some level of the example naming standards that SAP utilise in its pre-supplied integration content but also requirements to meet expected SRTs specific BTP-IS naming.

* Provide appropriate description to BTP-IS development objects like filter, communication channels, content modifier etc. based on Apache camel case naming convention.
* The iFlow design and naming will be environment neutral and only the actual connection configuration/mapping will have environment and system specific values.
* The approach must be to externalize all parameters of the components in the integration iFlow whenever possible.

The table below summarizes the naming convention to be adopted in Client for BTP-IS development.

|  |  |  |  |
| --- | --- | --- | --- |
| ***Object Type*** | ***Definition*** | ***Convention Syntax*** | ***Example*** |
| **Custom Package Name** | A collection of related APIs or services, belonging to one product or product area, packaged and delivered together. | Name: The name of the package should refer to the two products plus product lines between which the integration needs to take place if it is point to point. If you are developing generic integration packages or country specific packages then refer to generic and country specific sections in the example. | **Point to Point:**  Name: SAP Hybris Cloud for Customer Integration with SAP Hybris Marketing.  If you are developing generic interfaces like EDI or API(S)  and you don’t want to tie IFLOWS to a specific system then use naming conventions like below.  **Generic:**  <Business Object/Domain Description>for<Sender or Reciever System>  **Ex:**EDI Integration Templates for SAP Cloud Platform Integration Advisor  **Ex**: EDI Integration Templates for Successfactors  **Ex:** S4 HANA API(S)  **Country Specific**  If you are developing  package specific to country like tax interfaces then I would follow:  <Business Object/Domain Description> for <Country><Sender or Reciever System>  **Ex:**Payroll e-Filing of Employees’ Payments and Deductions for UK HMRC |
| **Point to Point:**  Technical Name: Z\_<Sending System Name>\_Integration\_ With\_<Receiving System Name>  **Generic:**  Z\_<Business Object/Domain Description><Sender or Reciever System>  **Country Specific:**  Z\_<Business Object/Domain Description><Country><Sender> OR/AND <Reciever System> | **Point to Point** :  Technical Name: Z\_Salesforce\_Integration\_With\_SAPC4HANA  **Generic:**  Z\_SupplyChain\_S4HANAAPI  **Country Specific**  Z\_PAYROLL\_UK\_HMRC |
| Vendor : <Organisation Name> | Vendor Name: Wales & East Utilities |
| Version: The version number should always be 1.0.0 when the package is productionized and should be incremented by +1.0.0 or 1.1.0 or 1.0.1 for every transport based on whether change is major or minor or micro. | Version : 1.0.0(when productionised) ,1.0.1(After micro change of first transport of artefact) on whether change is major or minor or micro. |
| Overview: The full long description of the package describing the usage, functionality and goal of the package. | Overview: This integration package enables you to integrate business processes between SAP Hybris Marketing or S/4HANA Cloud Marketing Edition and SAP Hybris Cloud for Customer. The integration scope includes a call center scenario, the creation of leads in SAP Cloud for Customer from a campaign in SAP Hybris Marketing and the replication of accounts, contacts, individual customers, leads and opportunities from SAP Hybris Cloud for Customer to SAP Hybris Marketing. |
| TAGS: The package should be tagged with country and line of business and industry using relevant dropdowns and a common keyword search should be based on most commonly used in package short description or Interface ID(S) of package. | Country:UK Line of Business: Metering Industry : Oil and Gas Keyword : Customer, NI000001 Product : SAP Marketing CLOUD, SAP C/4 HANA Project Name : “Wales and West Metering DTE |
| Documents: The API / Interface Documentation should be linked for each interface/artefact in the package and name should match exactly with IFLOW Name. | Create Planning Orders |
| **iFlow** | Integration flow is a BPMN-based model that is executable by orchestration middleware | <Interface Action i.eSend/Recieve/Create/Publish/Update/Delete/Replicate/Insert/Maintain><Interface Business Object ShortDescription> | Create Planning Orders |
| **Content Modifier** | This Step is used to modify content | Modify\_<Details of modification> | Modify\_logHTTPHeader |
| **Script** | This step is used to create groovy script to handle complex flows | SC\_<Objective of script> | SC\_formatMessage |
| **Message Mapping** | Message Mapping enables to map source message to target message | MM\_<SourceMessage>\_to\_<TargetMessage> | MM\_MATMAS\_to\_ProductCategoryReplication |
| **Sender** | Sender Endpoint | S\_<Sender Partner> | S\_C4HANA |
| **Receiver** | Receiver Endpoint | R\_<Receiver Partner> | R\_SIEBEL |
| **Value Mapping** | Value Mapping is used to map source system values to target system values. | VM\_<Source Agency>\_to\_<Target Agency>\_<FieldName> | VM\_C4HANA\_to\_ HYBRISMARKETING\_SALUTATION |
| **Communication Channel** | Communication Channel is used to convert Source message protocols into target message protocols | < Adapter Type>\_<Sender/Receiver>\_<BusinessSystem>\_<Interface Short Description> | ODATA\_Marketing Cloud\_C4HANA\_BusinessPartnerReplication |
| **Local Process Flow** | Local Process flows are used to modularise the complex integration flow | <Description of the objective that local process flow is designed to achieve> | Successf |

## Coding

SAP Cloud Platform Integration supports two forms of scripting languages: Groovy and JavaScript  
All code should be easily readable, to ease future maintenance and extension. Coding standards and guidelines are listed as key items to follow but there is no automated tool to enforce or assist in apply these guidelines, instead individuals will need to adhere to these standards and peer QA reviews will occur to ensure adoption. Key items to take into account are

* Code indentation. Must use standard 4 space.
* Code to follow typical best practices in terms of case/naming/patterns
* Comments: inline comments are not recommended. Code should be self-explanatory. Otherwise refactor your code until you can read it without a comment.
* Comments only exceptional cases.
* Code to be well structured, organized, loosely coupled and cohesive, use of objects/methods should be consistent – should follow object-oriented programming language methodology.

## Script Guidelines

Points to consider while choosing XSLT or Scripts

* There are no straight guidelines on using one over the other – it depends on the use case.

You may consider the following points while planning:

* If you need to change the XML structure, like cloning a node or deleting a node, etc., it is easy and quick to develop it using XSLT.
* If you need to perform some calculations based on the XML data and set some variables, an optimal instrument would be a script.
* XML parsing within a script is susceptible to XXE attacks (XML External Entity attacks) - instead request a specific payload type as this will trigger type conversion implicitly. This will not be prone to any attacks as the parser that is running during that conversion is secure. For example, instead of requesting the message body as an Input Stream and then parsing it into a Document, you can request the message body as a Document.
* Datatype changes can prove to be very expensive and may bring down the performance. Try to keep the conversions to the minimum – e.g., if the message is the output of an XSLT as a DOM Document, converting it to Input Stream and then parsing it into DOM will cause quite some overhead.
* Do not convert an XML to string and perform string operations to manipulate the XML. Make use of parsers within the script.
* XSLT makes use of the Saxon parser, so whatever holds good for Saxon over SAX, STAX or DOM parsers also apply here.
* Consider the parser capabilities and footprint before choosing a DOM parser over SAX/STAX-
* Although it may be easy DOM parsing the XML in the script, it is important to note that DOM parsers are memory intensive as they load the entire XML in memory and create the XML tree from it. This is an important consideration especially when working with big datasets as they can cause an out-of-memory situation. However, if the dataset is not huge, DOM parsing can prove to be more performant. DOM parsing can also be a better choice when you need to parse the XML back and forth.

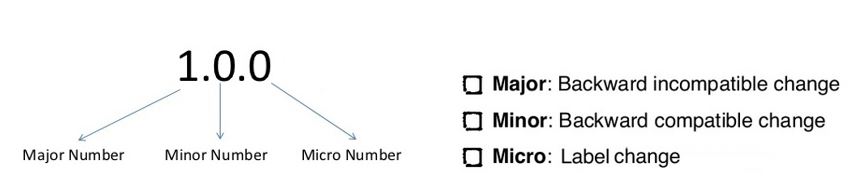
Note: SAX/STAX are very helpful when working with huge datasets as they stream the XML and do not load the entire XML in memory. Moving the XML back and forth may be expensive with these parsers.

## Source Control

BTP-IS Source Control

**Versioning**  
Versioning is the creation and management of multiple releases of a package, artifact hereunder Integration Flows (iFlows), all of which have the same general function but are improved, upgraded or customized.  
**Packages**  
Packages can be versioned loosely with just an input field. Before transportation it’s needed to input a version number of the package to verify that all meta data has been filled such as Description and Details.  
The versioning numbering will follow the approach of XX i.e., 01, 02 and so forth.  
**Artifacts**  
All artifacts in BTP-IS has versions following the standards of x.x.x as defined below wherein it’s possible to verify whether or not an object is backward compatible.

* Version change from 2.x.y to 2.(x+1).0: (e.g., 2.5.2 to 2.6.0) – a feature release  
  Such a version will contain new features compared to the previous version that will have an extended scenario – or simplification of the configuration or scenario.
* Version change from 2.x.y to 2.x.(y+1): (e.g., 2.6.0 to 2.6.1) – a patch release  
  Such a version will contain small patches and no new feature. I.e. a bug fix in the mapping.  
  Version 2.x.y.z: (e.g. 2.5.0.1) – a release containing late error fixes detected in last test cycle  
  Such a version will not happen too often and might only affect very specific small details.
* Version 3.x.y: A change of the major version will only happen, if an incompatible to the iFlow is implemented – i.e., massive changes to data definitions or mappings. These version changes should happen very rarely and only if massive changes to the flow is implemented.



All objects in BTP-IS should follow the same versioning concept for released versions. Only versioned objects are allowed to be transported. Until the object is to be formally unit tested, it’s allowed to keep the version without version number called Draft.

## Unit / Link Testing

1. Unit/ link testing is covered under testing strategy and Functional specification document. <consider Juit, Aunit, Postman etc.>
2. <JIRA with X-Ray plugin> will be used for unit testing and link testing (or relevant current agree test tool). It will be used to build manual and (when approved and available) automated test scripts.
3. <JIRA> will be used for defect analysis and reporting tool (or relevant current agreed defect management tool).
4. SAP Best practices can be used as reference to build test scripts. SAP provides test scripts and process flow diagram as part of standard integration content and can be used as a basis and/or check list to compare to when building the Unit Test scripts (if SAP content is applicable and available).

## Testing with S4 environment:

Artifacts will be moved from development system to quality system via standard transport/deployment mechanism agreed with SRT and it will be governed by the transport team.

## Quality assurance & Review

* The document artifacts (Functional and Technical specifications for example) will follow pre-agreed formats/guidelines and there is QA review templates to be completed by the 'author', peer and overall manager/QA lead.
* Developments will also have QA check lists to be completed by author, peer and Manager/QA Lead

## Security

All security related designs/tools need to be agreed with InfoSec team in SRT. The Digital Platform team will ensure the overall Technical Architecture adheres with InfoSec's requirements and this will also influence the design and thus standards for BTP-IS integration. Key points on security are   
Tenant Security  
The integration platform is designed as a cluster of virtual machines (nodes) that runs within the SAP cloud. Although all users that connect to the platform through the Internet share the same physical infrastructure.  
Tenants are strictly isolated from each other with regard to their resources (such as CPU and memory) and to the data that is processed on them. SAP stores the SRT data in a separate database schema   
Example: Tenants   
(DEV - <https://e1nnnnntmn.hci.eu1.hana.ondemand.com/itspaces/shell/discover> where the tenant ID is e1nnnnn)   
(QA: <https://ennnnntmn.hci.eu1.hana.ondemand.com/itspaces/shell/discover> where the tenant ID is ennnnn)   
Communication: The technical infrastructure comprises a set of technical components that can communicate with each other and with remote components in a secure way based on certain protocols such as HTTPS or SFTP, for example. In addition, user access to the technical infrastructure is designed in such a way that only users with well-defined permissions can access the different segments.  
All communication within the SRT network needs to be secure.

Data Management:  
Any data transmitted needs to be via secure methods.  
Secure File transmission can be achieved using SFTP or for direct communications either via a trusted HTTPS connection (Client trusts the root certificate of the server) or SNC for gateway communications.  
Authentication must also be addressed, e.g., in Connections from S/4 inbound to BTP-IS (HTTPS) authorization and authentication can be either User Role based or Client Certificate  
User Role  
Basic Client-Certificate with certificate-to-user mapping OAuth Client Credentials Grant OAuth SAML Bearer Destination  
OR  
Client Certificate (The most Advisable one)  
Client-Certificate without certificate-to-user mapping  
Interface/Service accounts need to be setup in BTP-IS that have the ability to process requests, for example the SAP system will set up accounts within the S/4 system for generic 'interface' processing (i.e., common access/functions plus these would be granted specific Functional access to the relevant business processing functions, data and actions. These system accounts would not be able to be used by end users (and when possible be restricted to system access rather than online user access).  
User creation and roles are covered under.

User Authorization and Roles:

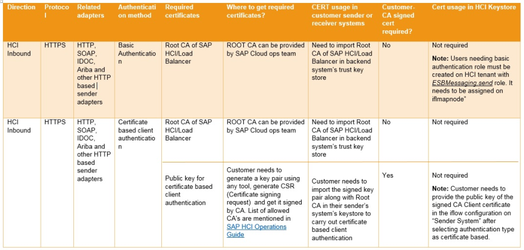
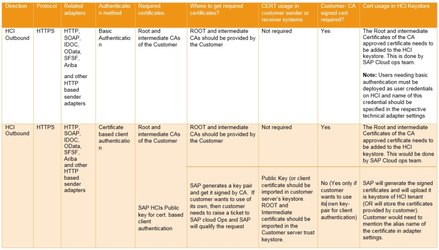
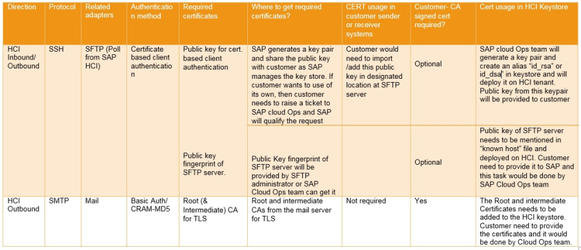
* User role and authorization strategy is covered under security and authorization strategy document.
* Developer, Administrator and Monitoring groups are created. Users are added under groups based on their role.
* Only users with administrator role can edit groups and assign roles.
* By default, BTP-IS used SAP cloud identity provider.

Tenant Provisioning:

* SAP is responsible for tenant provisioning.
* We can raise incident with SAP on Component LOD-HCI-PI-OPS for security and tenant provisioning related information.
* SAP is creating the user (S ID) but the appropriate Roles should be provided by the SRT local Admin team.

Transport Level Security

* SAP recommendation is to use CA signed certificates. Self-signed certificates cannot be used in BTP-IS. Detail list for CA can be found in best practice explorer.
* SAP / SRT BTP-IS Administrator manage key stores and known host file for Customers.
* SAP uses cloud connector as reverse proxy and secure tunnel between cloud services and on premise S4 HANA system.
* As BTP-IS is SAP cloud platform service hence connection between BTP-IS & SCP is HTTP, but the connectivity between SCP and on-Premises S/4HANA system is HTTPS through SAP Cloud Connector.

Security Checklist for transport level security (sourced from SAP recommendation blog, click on image for link):   
  
  
  
Message Level Security

* Encryption – Decryption of the message is used to impose message level security. PGP encryption – decryption is supported.
* Data security – Payload is encoded for data security as per SAP best practices.

As per the SAP if any sensitive data (Ex: Bank Statements, Purchase Order, Sales Order etc )/GDPR is moving out or coming into SRT Network which are confidential then encoding of data is advised

* Self-signed certificates can be used for message level encryption and signing. SAP recommendation is to use CA signed certificates.

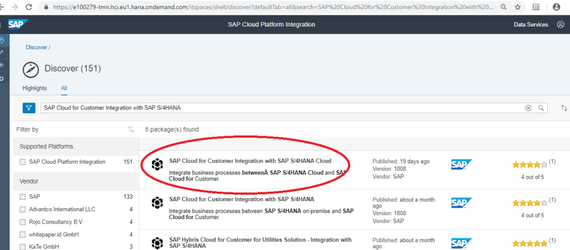
# Appendices

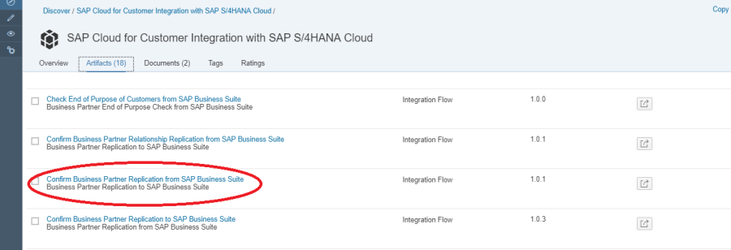
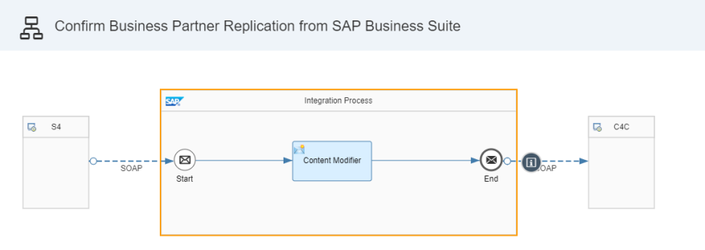
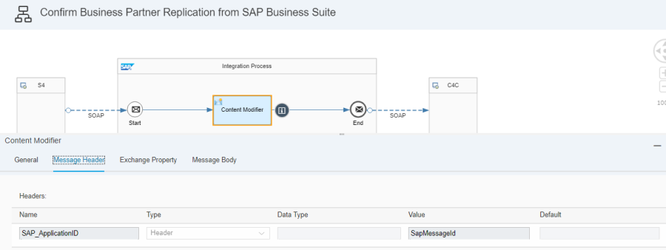
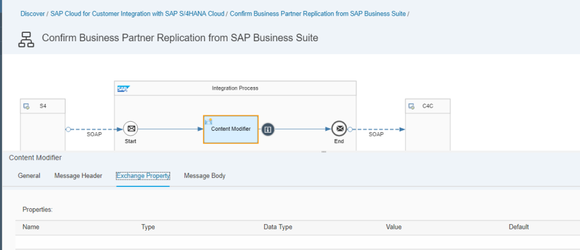
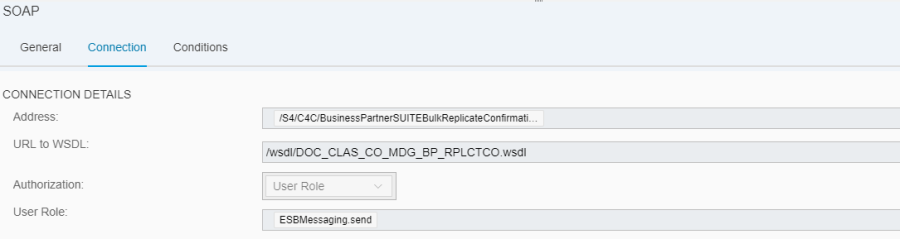
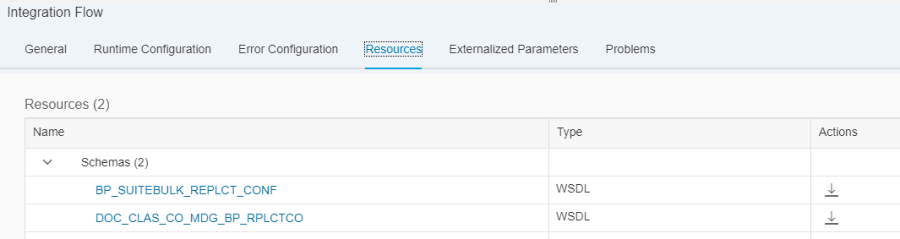
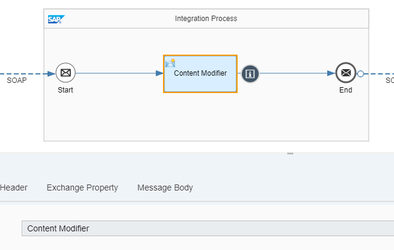
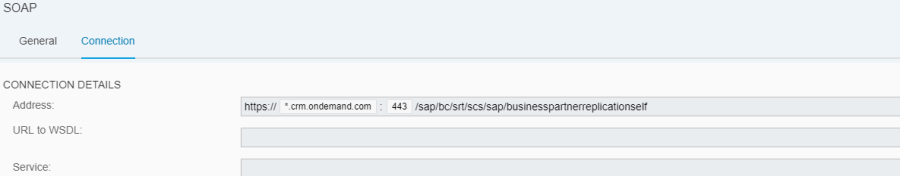
## Good practices and guidance

Please follow the SAP recommended and SRT documents for good practices around integration as per references/links at start of the document.

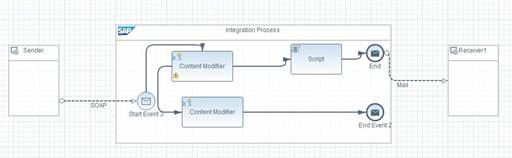
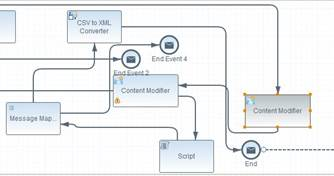
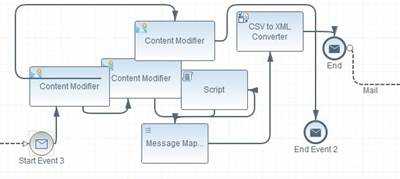
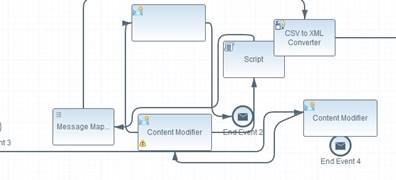
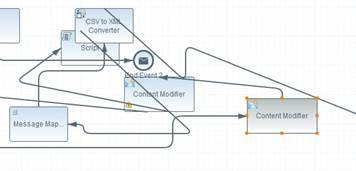
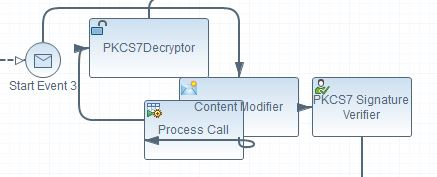
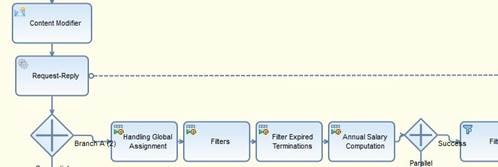
* SAP CIO guide around hybrid integration
* SAP Best practices explorer
* API Business Hub
* High Level Architecture document.
* Integration Approach/Strategy document.

## Example solutions / templates

SAP provides standard integration packages which are available in the respective SRT BTP-IS tenant.   
  
Pre-packaged Content   
  
Standard iFlow

  
An example of one standard integration scenario "**Confirm Business Partner Replication from SAP Business Suite**" which is available in the "SAP Cloud for Customer Integration with SAP S/4HANA" content   
  
  
  
In this scenario S/4HANA is sending the Business Partner Replication confirmation message to C4C.  
Connection details are maintained from S4 to BTP-IS as per the screen shot.   
  
Standard WSDL have been used.   
  
  
  
Within the iFLow one content modifier has been used.   
  
And on the receiver side SOAP adapter has been used to connect with C4C with the connection details given as per the screen shot.   
  
  
Hence, we can use the Standard integration content in the business scenarios and develop objects without many difficulties.

## Content design guidance

Following a standard in content layout design fosters ease of maintenance. Note that these diagrams are shown as examples and not to be followed as copy into the SRTs BTP-IS interface development.   
Always keep the flow direction from left to right. The sender always comes on the left and the receiver on right.   
Try to avoid overlapping sequence flows  
  
Avoid kinks (or confusing twist/turns) in the sequence and message flow connectors – try to keep them as straight as possible.  
  
  
  
Avoid overlapping process steps – in case you need many process steps, try expanding the canvas and arrange the process steps neatly.  
  
Modularize wherever possible – in case of complex logics, try to break it down into small, easy to understand modules. Move the logic of each module into a sub-process. Name the sub-process appropriately to describe the module's operation.  
  
   
Do not mix multiple transformations in a single script or sub-process – one sub-process should only contain the logic for one function.  
Un deploy unwanted and unused flow from your tenant.  
Do not assign the whole XML message to a header or a property unless necessary. Clear it once done.