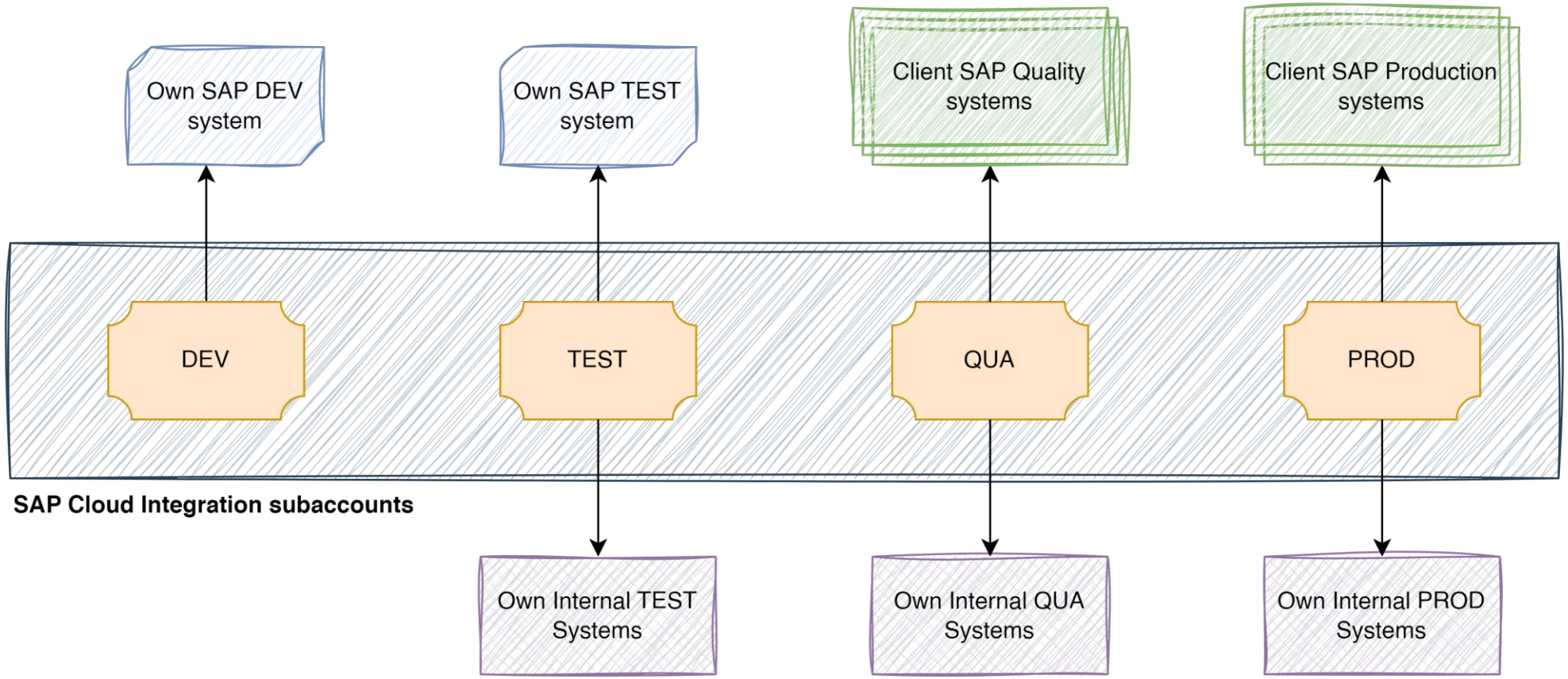
Git-enabled SAP Cloud Integration development process

Prepared by: Marat Bareev

Before git

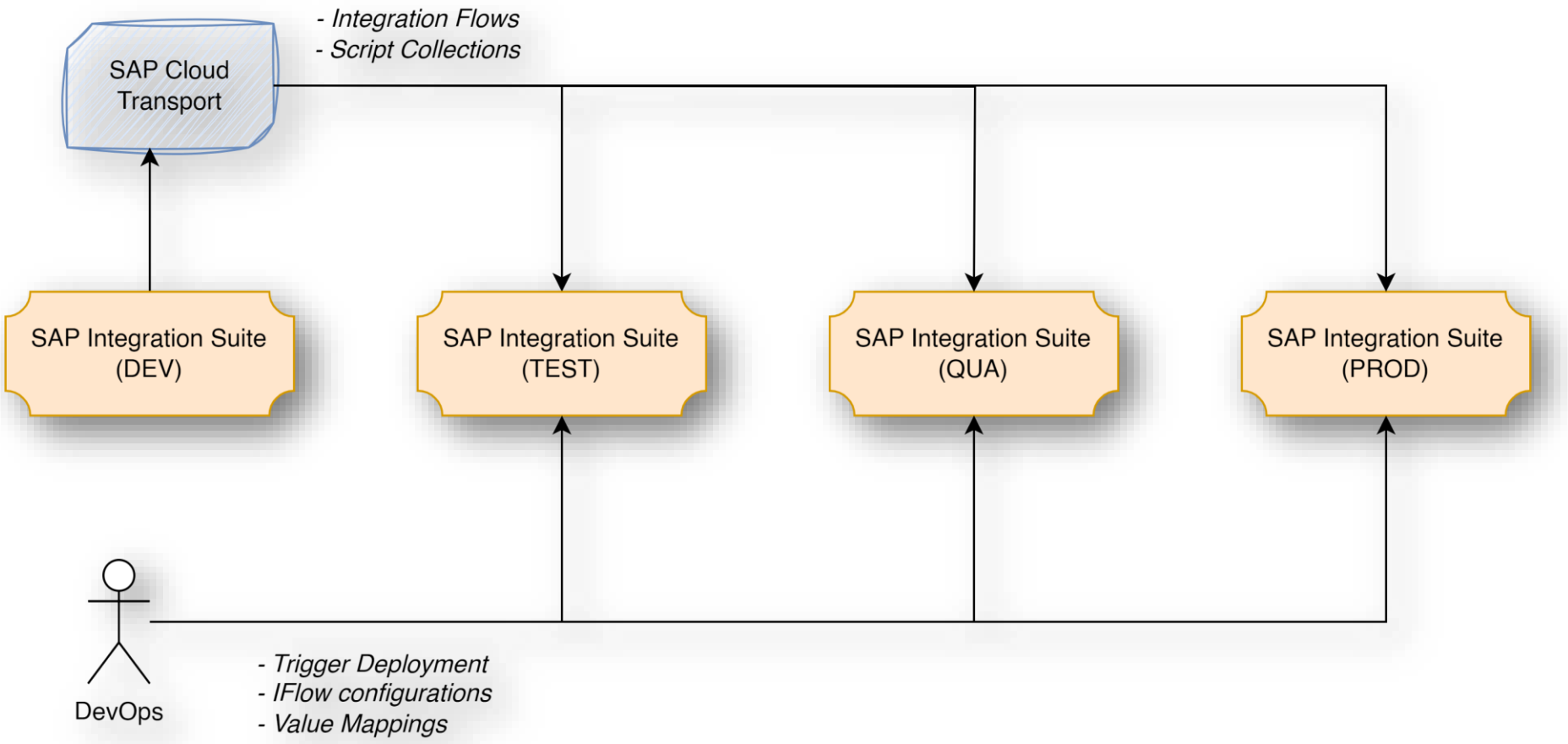
Integration landscape



Before git

Used methods of transport:

1. Manual edit (value mappings only)
2. SAP Cloud Transport (integration flows, script collections)



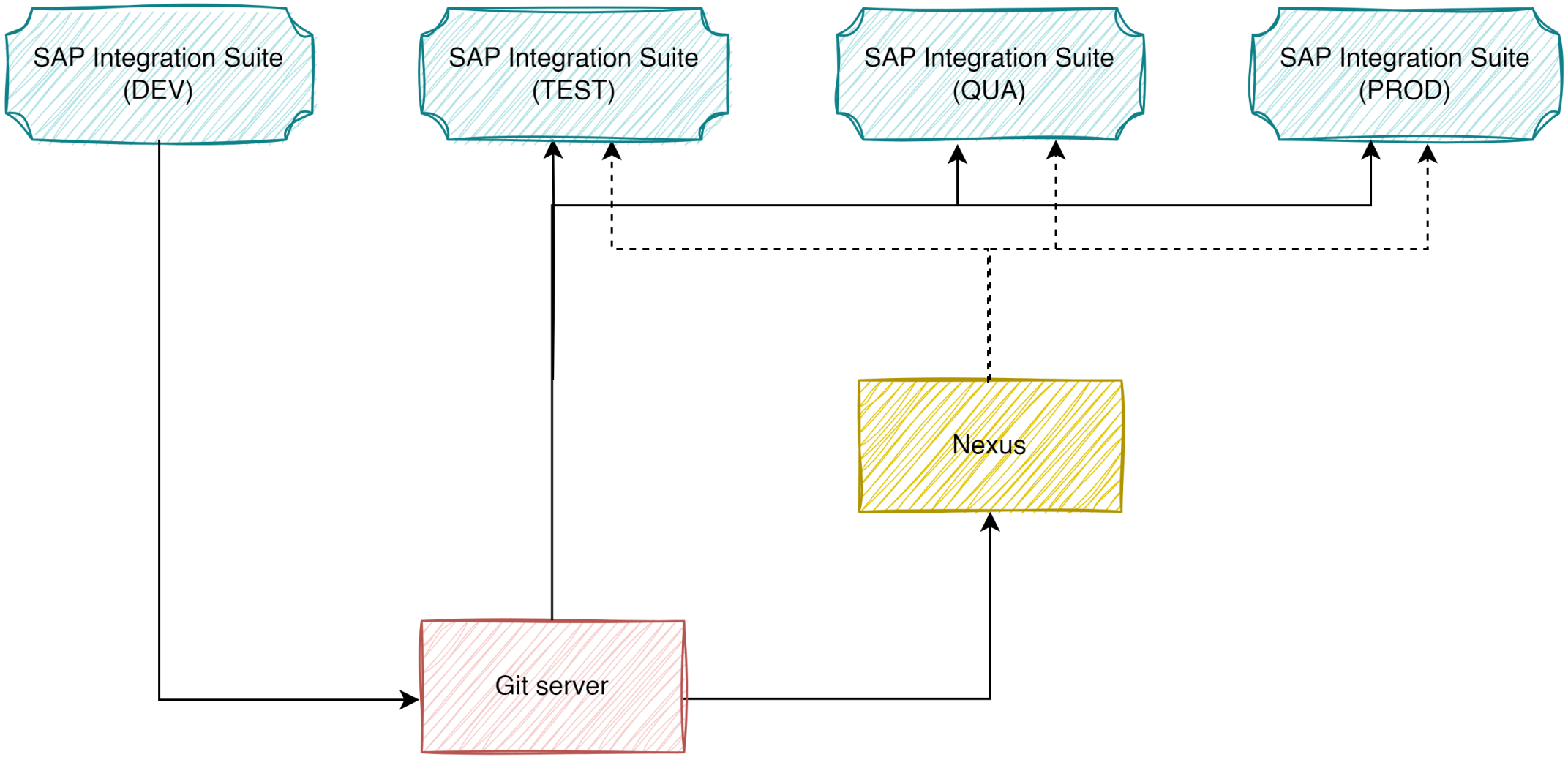
Before git

Problems with initial process:

1. Tedious value mappings update process which must be done manually
2. After import from SAP Cloud Transport manual deployment is required
3. After import from SAP Cloud Transport manual configuration of iflows is required
4. Artifacts from different packages cannot be transported in one transport request
5. Need to coordinate value mapping changes between developers
6. Need to coordinate integration flow configuration changes between developers
7. Need to know / remember if artifacts must not be deployed in some CPI environments

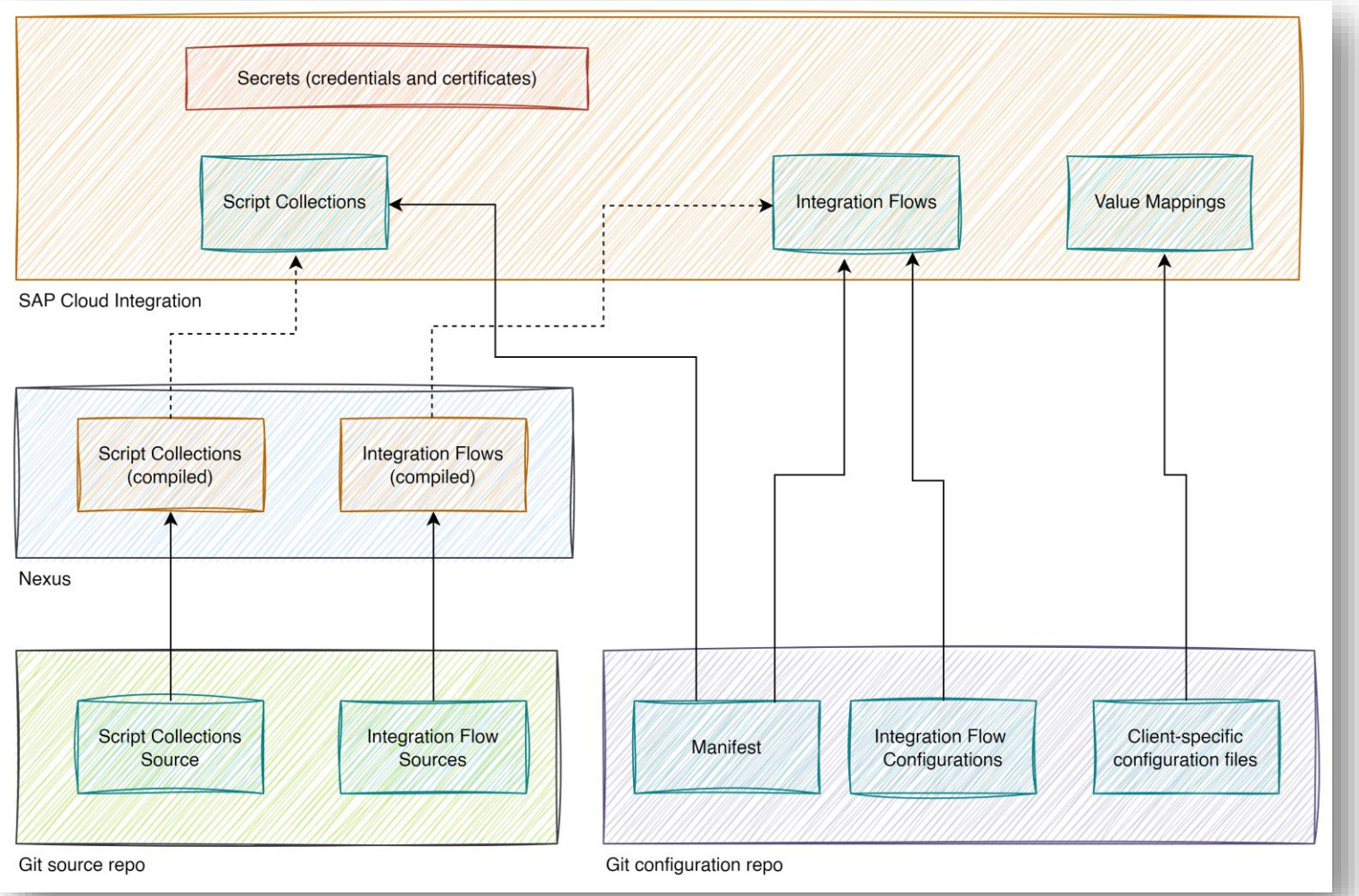
Introducing Git into process

Logical Transport landscape



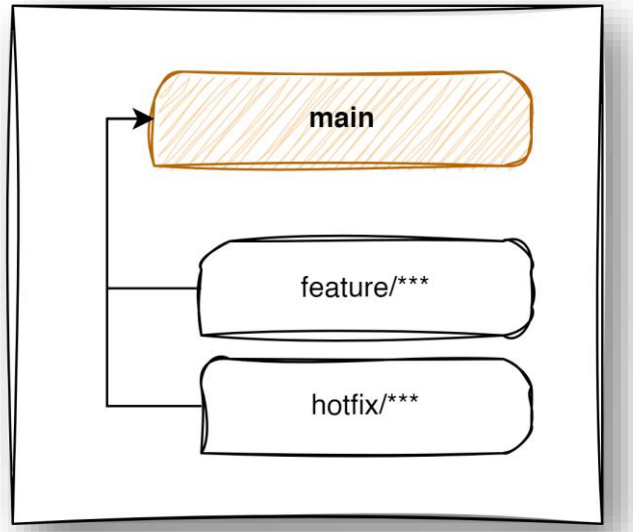
Git repositories

Git repositories and artifacts

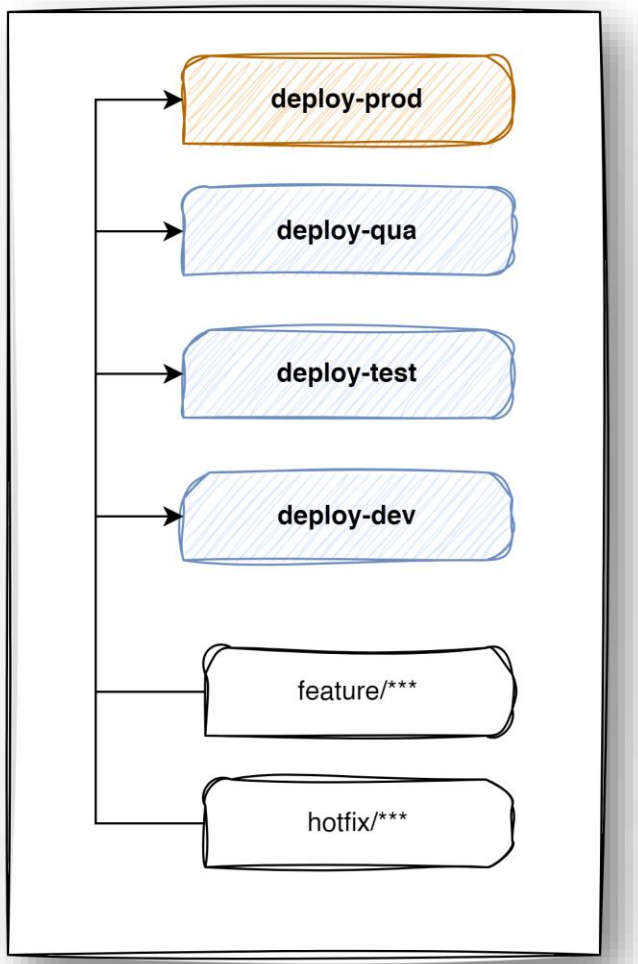


Git repositories

List of branches for Git repositories



Source repository

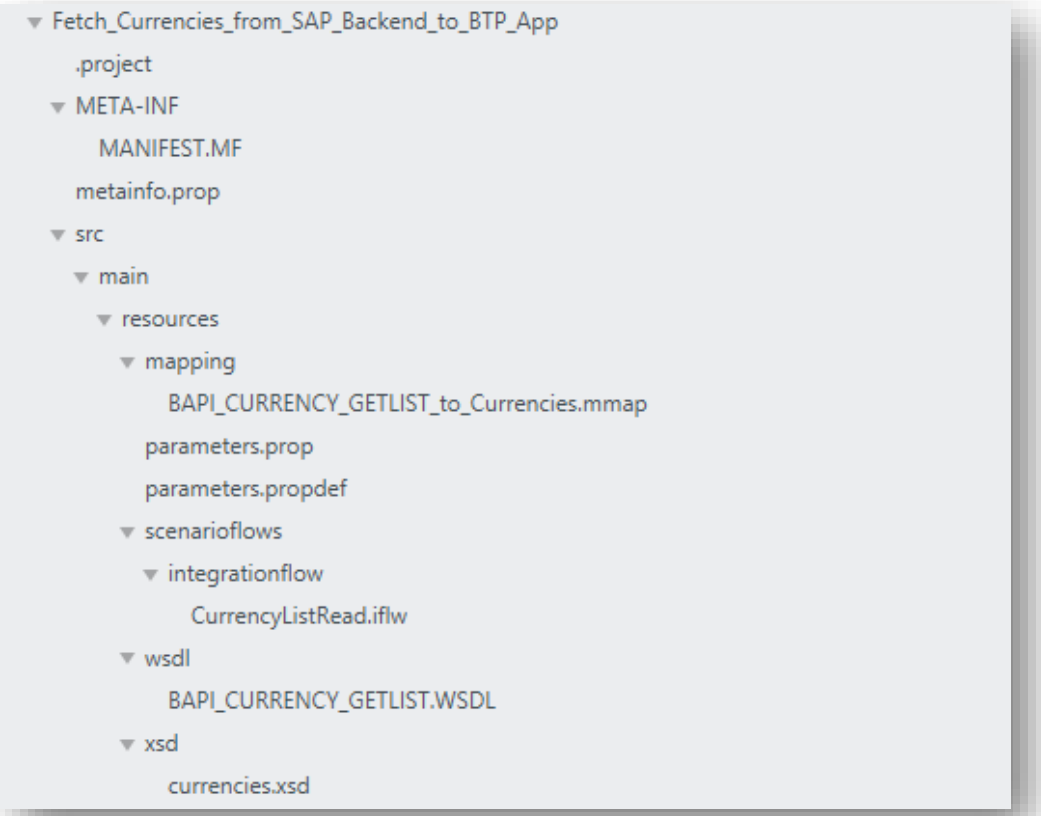


Configuration repository

Git repositories

Contents of Sources repository (1)

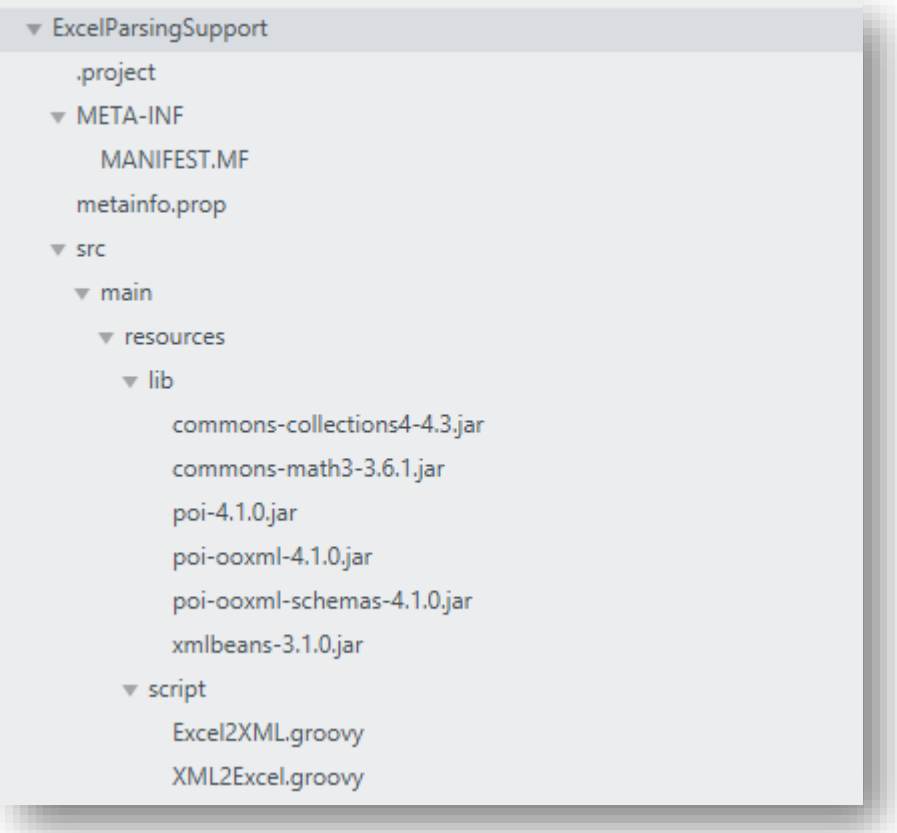
1. Example Integration Flow:



Git repositories

Contents of Sources repository (2)

2. Example Script Collection:



Git repositories

Contents of Configurations repository (1)

1. Example manifest contents (one per package)

```
iflow:Check_Company_Code_and_Customer_from_SAP_Backend:1.0.6
iflow:Check_Company_Code_and_Vendor_from_SAP_Backend:1.0.3
iflow:Fetch_Company_Codes_from_SAP_Backend:1.0.10
iflow:Fetch_Currencies_from_SAP_Backend_to_BTP_App:1.0.6
iflow:Fetch_Customers_and_Company_Codes_from_SAP_Backend_to_BTP_App:1.0.11
iflow:Fetch_Customers_from_SAP_Backend_to_BTP_App:1.0.5
iflow:Fetch_Document_Types_from_SAP_Backend_to_BTP_App:1.0.6
iflow:Fetch_FI_Document_Update_Status_from_SAP_Backend_to_BTP_App:1.0.4
iflow:Fetch_Payment_Blocks_from_SAP_Backend_to_BTP_App:1.0.9
iflow:Fetch_Vendors_and_Company_Codes_from_SAP_Backend_to_BTP_App:1.0.4
iflow:Send_IDOC_Directly_to_SAP_Backend:1.0.7
iflow:Update_SAP_Backend_FI_document:1.0.23
script:ExcelParsingSupport:1.0.8
```

Git repositories

Contents of Configurations repository (2)

2. Integration Flow configuration files (one per integration flow):

- CrossFunctionalities
  - Check_Company_Code_and_Customer_from_SAP_Backend.txt
  - Check_Company_Code_and_Vendor_from_SAP_Backend.txt
  - Fetch_Company_Codes_from_SAP_Backend.txt
  - Fetch_Currencies_from_SAP_Backend_to_BTP_App.txt
  - Fetch_Customers_and_Company_Codes_from_SAP_Backend_to_BTP_App.txt
  - Fetch_Customers_from_SAP_Backend_to_BTP_App.txt
  - Fetch_Document_Types_from_SAP_Backend_to_BTP_App.txt
  - Fetch_FI_Document_Update_Status_from_SAP_Backend_to_BTP_App.txt
  - Fetch_Payment_Blocks_from_SAP_Backend_to_BTP_App.txt
  - Fetch_Vendors_and_Company_Codes_from_SAP_Backend_to_BTP_App.txt
  - Send_IDOC_Directly_to_SAP_Backend.txt
  - Update_SAP_Backend_FI_document.txt

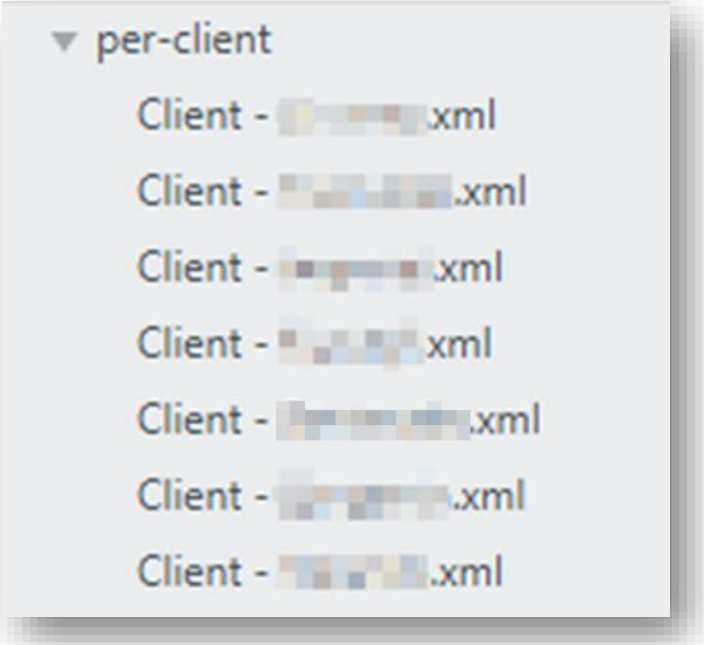
Example contents of configuration file:

```
xsd:string Address = [redacted]
xsd:string Directory = SAPBTPTEST/Outbox
xsd:string File Name Pattern = *.xlsx
xsd:string Private Key Alias = [redacted]
xsd:string ReceiverID = [redacted]
custom:schedule Scheduler = 0 0/5 0-23 ? * * * --tz=Etc/GMT
xsd:string SenderID = [redacted]
xsd:string User Name = [redacted]
```

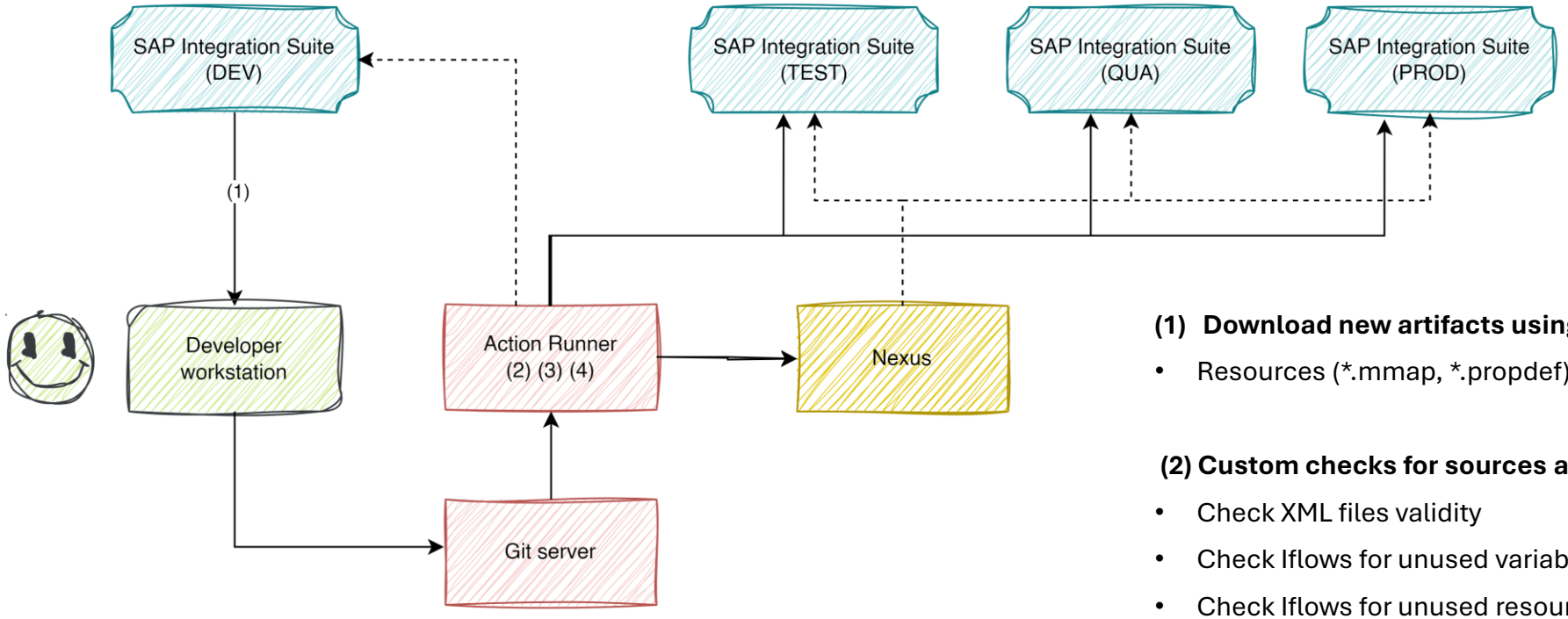
Git repositories

Contents of Configurations repository (3)

3. Client-specific XML configuration files



Development process (1)



(1) Download new artifacts using custom desktop tool.

- Resources (*.mmap, *.propdef) has XML formatting applied

(2) Custom checks for sources are performed:

- Check XML files validity
- Check Iflows for unused variables
- Check Iflows for unused resource files
- Check Groovy scripts for syntax, errors and validation warnings

(3) Configuration repository checks are performed:

- If target artifact version exists
- If configuration files match iflow available parameters

(4) Value Mappings are compiled from XML files:

- Validates if credentials and certificates exist in target environment

Development process (2)

Workflow scenarios for sources repository

Syntax check
Success

- Set up job
- Checkout code
- Prepare BaseX
- Check XML files
- Check IFlows for unused external parameters
- Check IFlows for unused resource files
- Run Groovy syntax check
- Complete job

Build and publish artifacts [MASTER]
Success

- Set up job
- Checkout code
- Prepare BaseX
- Compile manifests
- Attach manifests
- Validate manifests
- Build artifacts
- Publish artifacts to Gitea
- Complete job

Development process (3)

Workflow scenarios for configurations repository

Verify artifacts
Failure

- Set up job
- Checkout code
- Prepare BaseX
- Download published artifacts
- Validate iflow variables
- Complete job

Deploy artifacts to target environment
Failure

- Set up job
- Set branch-specific variables
- Checkout code
- Prepare BaseX
- Download published artifacts from Git
- Validate iflow variables
- Pull manifests and iflow configs from SAP Cloud Integration
- Backup current manifests to GIT
- Backup current iflow config to GIT
- Upload artifacts to SAP Cloud Integration
- Apply iflow configs SAP Cloud Integration
- Trigger artifacts deployment SAP Cloud Integration
- Complete job

Deploy client config files
Success

- Set up job
- Set branch-specific variables
- Checkout code
- Prepare BaseX
- Compile client configurations
- Upload artifacts to GIT
- Validate references against target environment
- Deploy artifacts to target environment
- Complete job

Pros and cons (1)

Direct pros:

- Auto-deploy artifacts from multiple packages at once
- Auto-validate credentials and certificates in target environment
- Auto-apply environment-specific configurations
- Generate complex value mapping based on client specific XML files
- Ability to plan configuration for all environments
- Validate source code (unused resources, unused variables, groovy code and xml validation + more coming)
- Can deploy artifacts only to certain environments
- Perform configuration values backup

Pros and cons (2)

Git-sponsored pros:

- Easy and clear change management via pull requests with multiple developers
- Embedded resources and source files are now fully searchable
- Configuration values across all environments are searchable
- Accidental (unintended) changes are visible
- All changes are easily reversible directly

Pros and cons (3)

Cons:

- SAP CPI DEV environment is still required as a starting point
- Draft versions of artifacts cannot be exported
- No change history available in SAP CPI TEST / QUA / PROD environments, only in Git
- This is a completely custom process (however solution relies solely on standard SAP CPI APIs)

Future development ideas

- Add secrets management (Vault or similar)
- Add source code security scan (groovy, java)
- Enable branch protection / approval process for PROD