

APACK

A Package and Dependency Manager for ABAP

Sebastian Wolf, Development Architect - Central Engineering, SAP SE

@Ygriega

May 2019

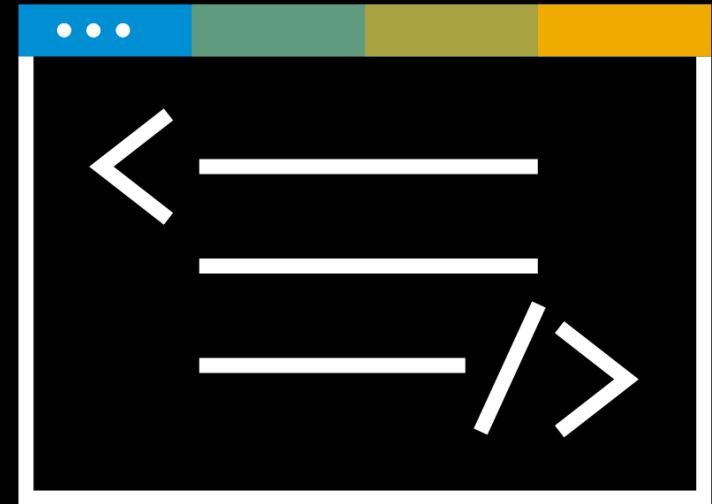
PUBLIC

THE BEST RUN



ABAP – A Special “Snowflake”

- ABAP: Over 30 years of history
- Considerable evolution especially in recent years
- However: Special characteristics have remained
 - In-system development
 - No local deployment/development
 - Full-fledged ABAP instance required for development
 - System applies two object states: Active/Inactive
 - System-central, proprietary version control system
 - No explicit dependency management
 - Weak, but complex package management
- abapGit to the rescue?!
 - Community-driven distributed version control for ABAP
 - Adopted in SAP Cloud Platform



ABAP on SAP Cloud Platform (Steampunk) – State-of-the-Art

- Custom code can be imported via abapGit
- Artifact transport between instances with gCTS

Whitelist Implications

- Custom code can only use whitelisted objects
- Whitelist is still very restricted
- → Ports of existing solutions and new developments are challenging

Porting existing foundation libraries

- Re-use libraries will need to be built
- Alternative: Port/Adopt re-use functionality from not-whitelisted components
- Required libraries can currently only be referenced via README
→ **Very error-prone and cumbersome!**



Wishes and Questions by the abapGit Community

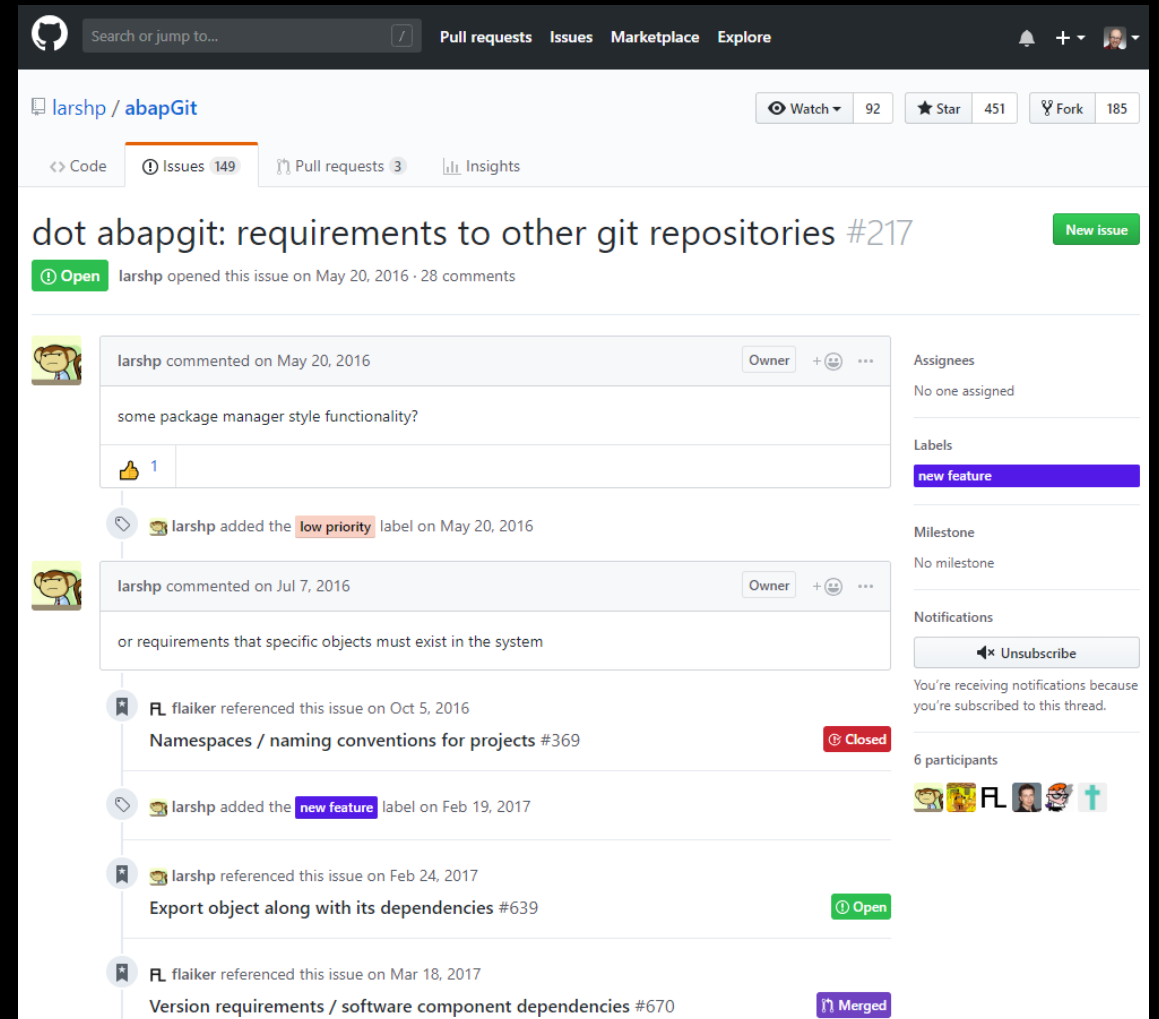
- <https://github.com/larshp/abapGit/issues/217>
- Discussed since May 2016 (maybe longer...)

Requirements

- Deploy projects with all dependencies!
- Support for forks, offline/ZIP-use
- Central repository like Maven Central
- Company-internal repositories required

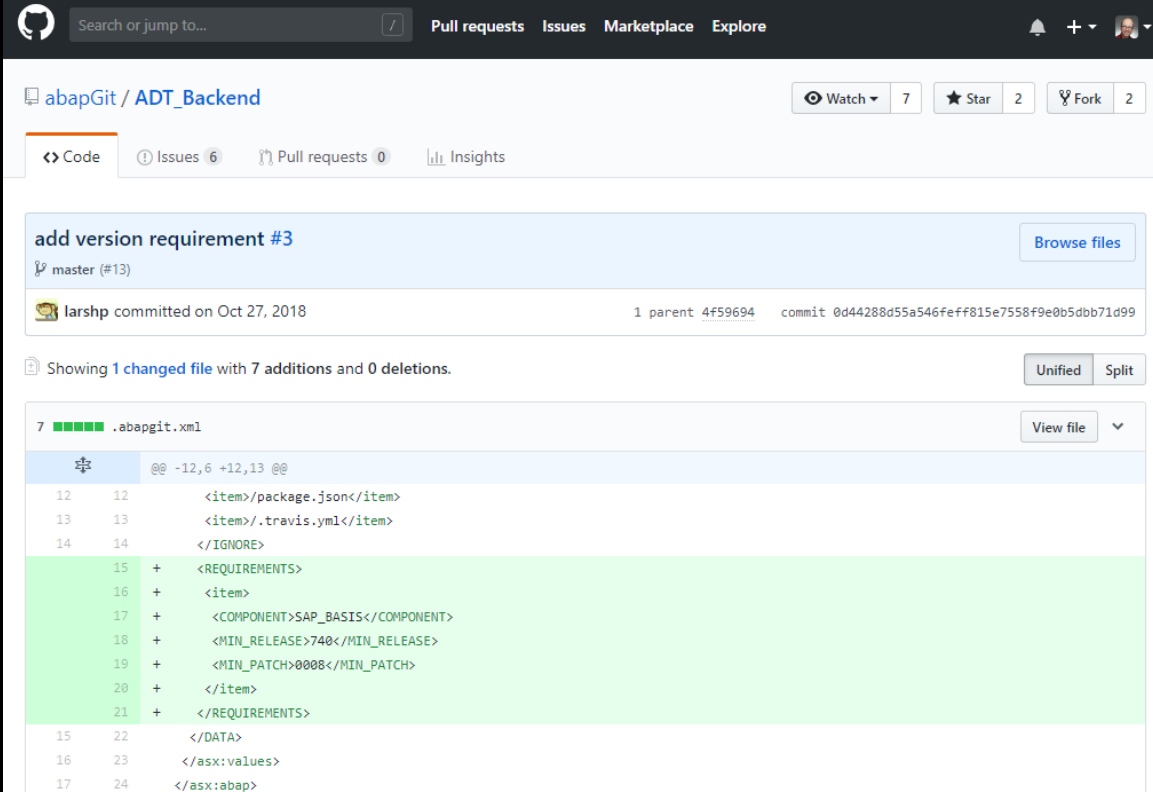
Questions

- Use the PINF (package interface) object type?
 - But what about system-local naming?
- Versioning or no versioning?
- Integrated in abapGit or independent project?



abapGit and Requirements – State-of-the-Art

- Requirements can be included optionally
- Coarse-grained requirement specification
 - Software Component
 - Minimum Release
 - Minimum Patchlevel/Service Pack
- Tight integration in abapGit



abapGit / ADT_Backend

add version requirement #3

larshp committed on Oct 27, 2018

Showing 1 changed file with 7 additions and 0 deletions.

Unified Split

7 .abapgit.xml

```
@@ -12,6 +12,13 @@
12 12      <item>/package.json</item>
13 13      <item>/travis.yml</item>
14 14      </IGNORE>
15 15      + <REQUIREMENTS>
16 16      + <item>
17 17      +   <COMPONENT>SAP_BASIS</COMPONENT>
18 18      +   <MIN_RELEASE>740</MIN_RELEASE>
19 19      +   <MIN_PATCH>0008</MIN_PATCH>
20 20      + </item>
21 21      + </REQUIREMENTS>
15 22      </DATA>
16 23      </asx:values>
17 24      </asx:abap>
```

Let's welcome APACK!

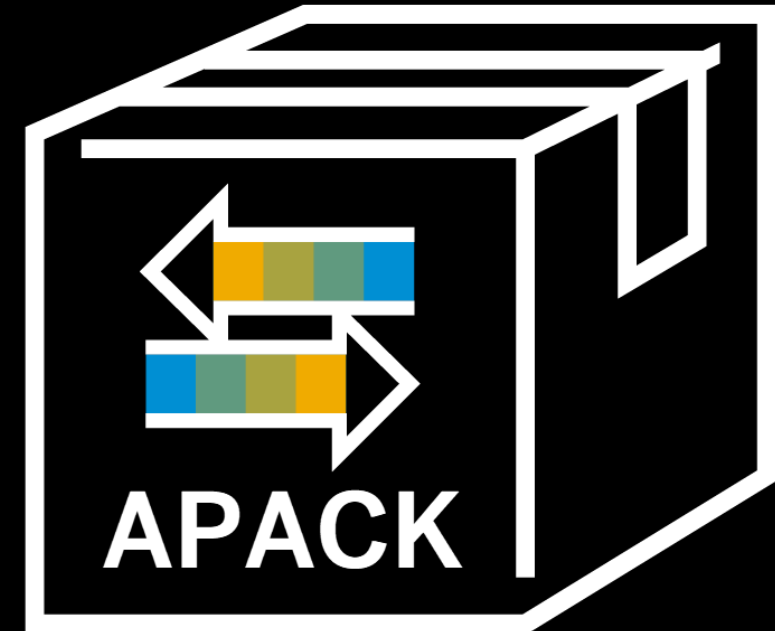
- Independent of ABAP package concept (no PINF, DEVC...) – No issues with system-local names/missing global naming conventions
- Package definition for re-use (group ID, artifact ID, version)
- Explicit dependencies between re-use packages
- Fully compatible with abapGit, but designed independently

Rationale

- Steampunk whitelist might not grow fast enough
- Enable Steampunk users to develop and re-use „libraries“
- Nurture ABAP open-source community (dotabap.org)

Implementation – Status Quo

- Import Steampunk-only, integration in abapGit plugin for ADT
- Export On-Premise-only, integration in standard abapGit
- No versioning







The APACK Manifest – ABAP implementation


- Marker interface IF_APACK_MANIFEST
- One class in a re-use library package must implement it
- Contains single descriptor field
 - Group ID
 - Artifact ID
 - Version
 - Repository Type (currently only ,abapGit‘ and added automatically)
 - Git URL
 - Dependencies (Group ID, Artifact ID, Git URL)

```
1  CLASS zcl_jak_apack_manifest DEFINITION
2  PUBLIC
3  FINAL
4  CREATE PUBLIC .
5
6  PUBLIC SECTION.
7  INTERFACES: if_apack_manifest.
8  METHODS: constructor.
9  PROTECTED SECTION.
10 PRIVATE SECTION.
11 ENDCLASS.
12
13
14 CLASS zcl_jak_apack_manifest IMPLEMENTATION.
15 METHOD constructor.
16   if_apack_manifest~descriptor-group_id = 'sap.com'.
17   if_apack_manifest~descriptor-artifact_id = 'abap-platform-jak'.
18   if_apack_manifest~descriptor-version = '0.1'.
19   if_apack_manifest~descriptor-git_url = 'https://github.com/SAP/abap-platform-jak.git'.
20   if_apack_manifest~descriptor-dependencies = VALUE #( ( group_id      = 'sap.com'
21                                                         artifact_id = 'abap-platform-yy'
22                                                         git_url      = 'https://github.com/SAP/abap-platform-yy.git' ) ).
23 ENDMETHOD.
24
25 ENDCLASS.
```

The APACK Manifest – abapGit Serialization

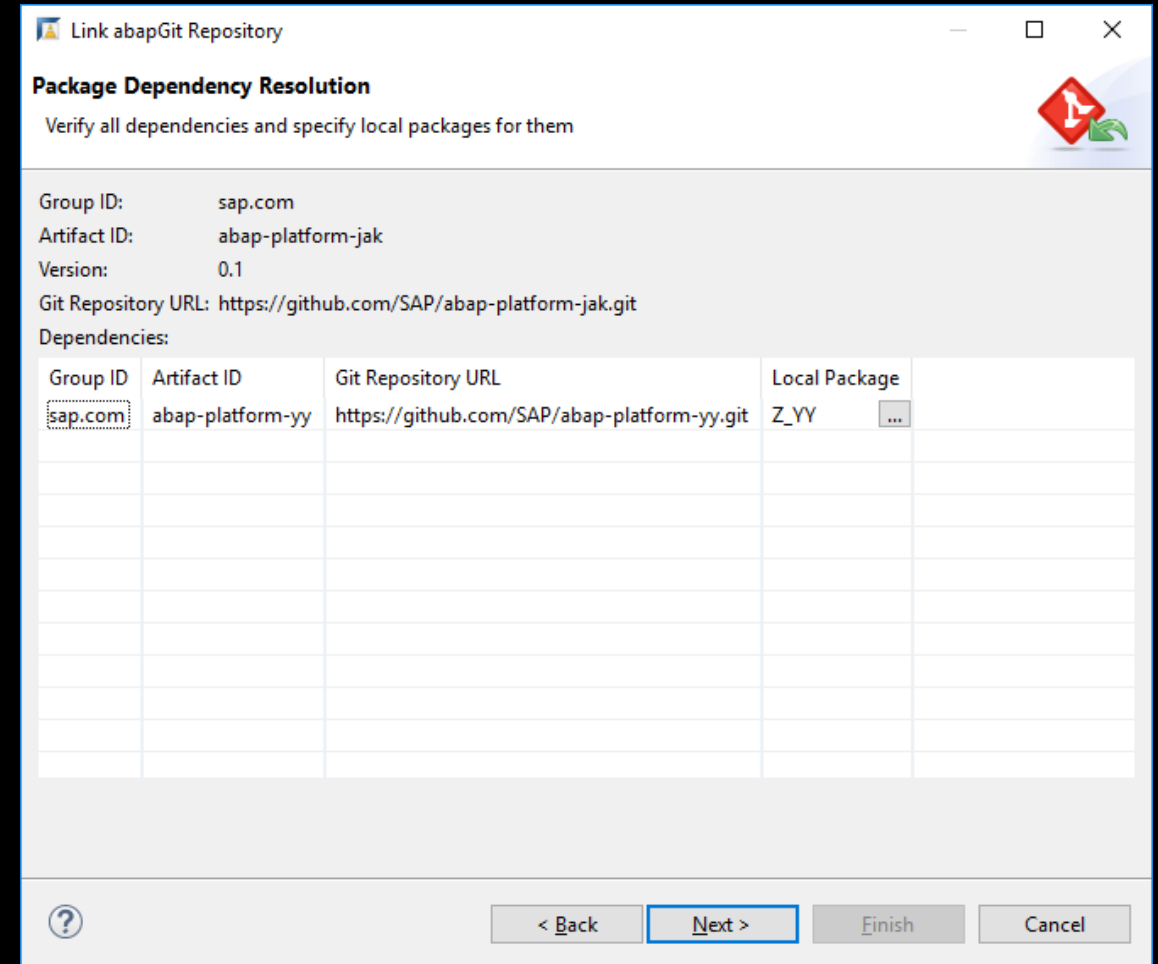
- New descriptor file `.apack-manifest.xml`
- Located on top level of abapGit project
- Automated serialization via abapGit extension

 SebastianWolf-SAP Add repository type to APACK manifest	
 <code>src</code>	Add repository type to APACK manifest
 <code>.abapgit.xml</code>	Initialize JAK Repository
 <code>.apack-manifest.xml</code>	Add repository type to APACK manifest

 SebastianWolf-SAP Add repository type to APACK manifest	
1 contributor	
20 lines (19 sloc) 568 Bytes	
1	<code><?xml version="1.0" encoding="utf-8"?></code>
2	<code><asx:abap xmlns:asx="http://www.sap.com/abapxml" version="1.0"></code>
3	<code><asx:values></code>
4	<code><DATA></code>
5	<code><GROUP_ID>sap.com</GROUP_ID></code>
6	<code><ARTIFACT_ID>jak</ARTIFACT_ID></code>
7	<code><VERSION>0.1</VERSION></code>
8	<code><REPOSITORY_TYPE>abapGit</REPOSITORY_TYPE></code>
9	<code><GIT_URL>https://github.com/SebastianWolf-SAP/jak.git</GIT_URL></code>
10	<code><DEPENDENCIES></code>
11	<code><item></code>
12	<code><GROUP_ID>sap.com</GROUP_ID></code>
13	<code><ARTIFACT_ID>yy</ARTIFACT_ID></code>
14	<code><GIT_URL>https://github.com/SebastianWolf-SAP/yy.git</GIT_URL></code>
15	<code></item></code>
16	<code></DEPENDENCIES></code>
17	<code></DATA></code>
18	<code></asx:values></code>
19	<code></asx:abap></code>

Package Dependency Resolution during abapGit Import

- If APACK manifest is detected, all dependencies are automatically resolved
- Dependency resolution is done transitively, also dependencies of dependencies are taken in
- Local packages for all dependencies can be selected on a single wizard page
- Dependencies are all linked at once at wizard completion



The screenshot shows a dialog box titled "Link abapGit Repository" with a sub-header "Package Dependency Resolution". Below the sub-header is the instruction "Verify all dependencies and specify local packages for them". The dialog displays the following information:

Group ID: sap.com
Artifact ID: abap-platform-jak
Version: 0.1
Git Repository URL: <https://github.com/SAP/abap-platform-jak.git>

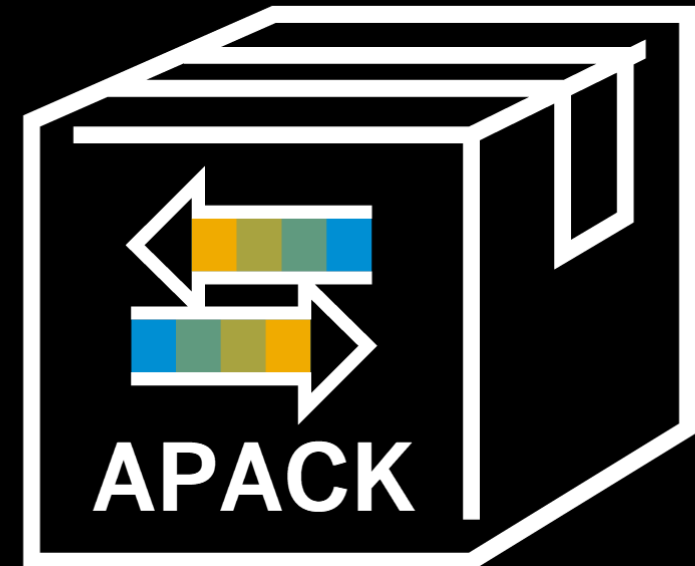
Below this information is a table titled "Dependencies:" with the following columns: Group ID, Artifact ID, Git Repository URL, and Local Package. The first row of the table is pre-filled with the following data:

Group ID	Artifact ID	Git Repository URL	Local Package
sap.com	abap-platform-yy	https://github.com/SAP/abap-platform-yy.git	Z_YY

At the bottom of the dialog, there are four buttons: "< Back", "Next >", "Finish", and "Cancel". The "Next >" button is highlighted with a blue border.

APACK: ABAP Dependency Management made easy and convenient

- ABAP gets an automated package and dependency management for re-use libraries and applications
- Easy-to-use and almost no overhead
- Direct integration with abapGit
- Open Source as abapGit
- Independent project which can be easily extended, e.g.:
 - Central repository for sources/packages
 - Company-specific repositories
 - Versioned dependencies
 - Ports to on-premise ABAP releases
 - ...
- Feedback and improvements welcome!
- HAVE FUN!



Thank you.

Contact information:

Sebastian Wolf

Development Architect – T&I Central Engineering

sebastian.wolf@sap.com

Twitter: @Ygriega, <https://twitter.com/ygriega>

GitHub: <https://github.com/SebastianWolf-SAP>