InvenioRDM - SWH SPECS

https://hedgedoc.softwareheritage.org/fc4e-wp6-t6-1-specs-template?edit

Component	*InvenioRDM software source code & metadata deposit*
Category	***RSAC***
Contact person	*Lars Holm Nielsen*
Email address	*lars.holm.nielsen@cern.ch*
Contributors	
Version	*1.0*
Date	*2022-11-25*

Overview

Integration of InvenioRDM (Scholarly Repository) with Software Heritage

Overall, the purpose is to develop tools and services for archival, reference, description and citation of research software artifacts, by implementing the key recommendations of the EOSC SIRS report to interconnect scholarly repositories with the Software Heritage universal source code archive, using the CodeMeta standard, and the Software Heritage intrinsic identifiers (SWHID)

Objectives

#	*Short description*
1	*Deposit in Software Heritage research software artifacts uploaded into InvenioRDM repositories, obtaining the corresponding SWHID*
2	*Expose SWHID in the artifact record maintained by InvenioRDM*
3	*Enable InvenioRDM to deposit and/or retrieve (curated) metadata of software artifacts from Software Heritage*
4	*Export citation information in one or more of the common open citation formats (BibLaTeX, CSL, codemeta.json)*
5	*Integrate and showcase the above into Zenodo*

Out of Scope

#	*Short description*
1	*Removal of content* *Software source code deposited in the repository may be subject to a take-down request or other requirements requiring the removal of the deposited material (e.g. legal reasons). The removal of content is assumed to be done independently for each service according to their policies. Thus there will be no automatic notification between the scholarly repository and the universal source code archive about removed content. The moment at which the content was transferred between the two services, the content was assumed to be public. This is consistent with how removal of content that might have been harvested by other repositories are dealt with.*
2	*Mixed deposit workflows*

Out of Scope	
	The deposit workflows outlined in UC1, UC2 and UC3 can be mixed up. For instance, *- V1: Deposit as a bundle (UC1)* *- V2: Deposit by enabling automatic deposit for a new release (UC2)* *- V3: Deposit by registering an already archived bundle (UC3)* *Each of the three versions leads to a new record in the scholarly repository which is linked to the other versions. The requirements specifically do not deal with how to transition a user between the different workflows.* *Similarly, a user may use the UC1 workflow to deposit multiple versions, and later figure out that an old release also has to be published.* *Another example, is a user depositing software source code and later decides to change it to restricted (with or without moderation).* *We specifically exclude dealing with these cases, and leave it up to the scholarly repository how it wants to deal with it according to its existing policies.*

Requirements

Overall, the requirements focus on researchers interacting with scholarly repositories, and how the repository interacts with the universal source code archive.

The user stories refer to the following roles:

- **Researcher**: An individual researcher or full research team.

User stories

#	*Short description*	*Priority*	*Feasibility*	*Reference*
A1	*As a researcher, I want to deposit bundle of source code into repository, so that it will be archived for the long term in the universal source code archive, SWH*	*H*	*4*	
A2	*As a researcher, I want to register already SWH-archived source code in repository, so that it will be recorded in the scholarly repository as well as in the universal source code archive, SWH*	*L*	*3*	
A3	*As a researcher, I want to deposit a new release of source code into a repository, so that it will be archived for the long*	*M*	*3*	

User stories				
	term in the universal source code archive, SWH			
R1	*As a researcher, I want to retrieve an identifier from the repository, so that it will reference the record and source code artifacts*	*H*	*5*	
D1	*As a researcher, I want to describe software using software specific metadata, so that the software is findable and reusable*	*H*	*4*	
D2	*As a researcher, I want to retrieve metadata export for deposit in repository, so that the metadata can be used in other workflows*	*M*	*4*	
D3	*As a researcher, I want to update metadata of software in repository, so that the metadata can be as accurate and complete as possible*	*L*	*4*	
C1	*As a researcher, I want to retrieve a citation export format from the repository, so that the software will be cited with correct attribution*	*H*	*4*	

- Priority: H=High, M=Medium, L=Low
- Feasibility: marking between 1 - 5, 5 is easy to implement and 1 is very difficult

Functional requirements			
#	*Short description*	*Priority*	*Reference*
UC1	**Deposit Bundle Through an Scholarly Repository** - A researcher explicitly deposits a software bundle (.tar.gz, .zip, etc.) and associated metadata into a scholarly repository. - *Optional:* The scholarly repository may implement a moderation mechanism to ensure a certain level of quality of the deposit (deduplication, affiliations of the team members, coherence of the metadata, etc.). - Once accepted, the bundle and metadata are archived in SWH, either immediately, or at the end of the optional embargo period. **Notes** - User stories A1 and A3 are considered as almost identical as both will produce a record in the scholarly repository (i.e. A3 does not modify an existing record, but creates a new record instead so that versions can be differentiated). - InvenioRDM can be configured with or without moderation support. The trigger to deposit the bundle	H	User stories: *A1, A3, R1, D1*

Functional requirements			
	in SWH is thus the public availability of a published record (i.e. either immediately or after an embargo period). *See annexe A: Sequence diagram*		
UC2	**Automated deposit of New Releases into Scholarly Repository and SWH (Manual and Automated)** Automated/manual deposit of new releases from a forge into a scholarly repository, as e.g. implemented in the Zenodo-GitHub integration, is an extension to depositing a bundle into scholarly repository. - The scholarly repository is notified about a new software release either automatically by the forge, or manually by a researcher. - The scholarly repository automatically extracts the deposit bundle and associated metadata from the forge. - Continues as "UC1 Deposit Bundle Through an Scholarly Repository"	*M*	User stories: *A3*
UC3	**Registering Already Archived Software in an Scholarly Repository** A researcher may need to register an artifact that has been already archived in SWH in a scholarly repository. To avoid duplication of work, machine readable metadata contained in designated files in the source code should be used to prefill the metadata deposit form.	*L*	User stories: *A2*
UC4	**Updating Existing Metadata in an Scholarly Repository** A research team may also need to update the metadata registered in a scholarly repository about an already archived software artifact. Previous versions of the metadata record should be recorded together with information on who changed what, when, and why. The changes should be made public to the extent possible by the repository's policies. **Notes** - Previous versions of the metadata record should be recorded and available, together with information on who changed what, when, and why.	*L*	User stories: *D3*

- Priority: H=High, M=Medium, L=Low

Non-functional requirements			
#	Short description	*Priority*	*Reference*
1	**Personal data compliance** The scholarly repository and universal source code archive are owned by different legal entities, and thus any exchange of personal data must be clearly reflected in the scholarly repository's privacy policy. Overall, exchange of any personal data should be avoided, and if done must comply with legal requirements.	*H*	
2	**Metadata formats must support interoperability and software properties exchange** The scholarly repository and universal source code archive exchange metadata in both directions. The repository sends metadata to SWH, and the repository fetches metadata from SWH. The metadata formats used for the exchange must enable the software specific metadata to be exchanged as well as enable the metadata formats to be used for citation generation. Likewise the metadata may be registered/harvested by other discovery systems.	*H*	User stories: *D1, D2, C1*
3	**Reliable exchange of deposits between repository and SWH** The scholarly repository is responsible for ensuring software records are sent to the universal source code archive and must be able to determine if a software record has been published to SWH.	*M*	

- Priority: H=High, M=Medium, L=Low

Specifications

Functional specifications			
#	*Short description*	*Priority*	*Reference*
1	*Feature flag, configuration and documentation* *Configuration* *The Software Heritage integration in InvenioRDM will need to be an optional add-on as not all InvenioRDM instances may want to enable the integration, nor is an InvenioRDM instance even guaranteed to host software source code. Software Heritage further requires authenticated access, thus as minimal a system administrator needs to provide their SWH credentials to the InvenioRDM application.* *Documentation* *We will ship InvenioRDM with Software Heritage support directly integrated, but users will need to*	*H*	

Functional specifications			
	enable the integration similar to how users need to enable the DataCite integration for registering DOIs.		
2	Marking resource types as software source code A scholarly repository may host many different resource types, and only software source code should be sent to SWH. *Tag resource types* InvenioRDM needs to enable a system administrator to tag which deposit resource types that should be sent to SWH. *Only public records* Further, only public records with public files are sent to SWH.	H	
3	Software-specific metadata *Codemeta for interoperability* The [Codemeta standard](https://codemeta.github.io/terms/) based on Schema.org is supported by both SWH, Zenodo and InvenioRDM as a common metadata exchange format. Further, InvenioRDM registers DOI via DataCite and thus provided DataCite metadata format as an exchange format for repositories and aggregators *Supported terms* The following Codemeta terms have existing fields in InvenioRDM or requires only minor adaptions to be supported: - `downloadUrl` - `fileSize` - `releaseNotes` - `author` - `citation` - `contributor` - `copyrightHolder` - `copyrightYear` - `creator` - `dateCreated` - `dateModified`	H	

Functional specifications

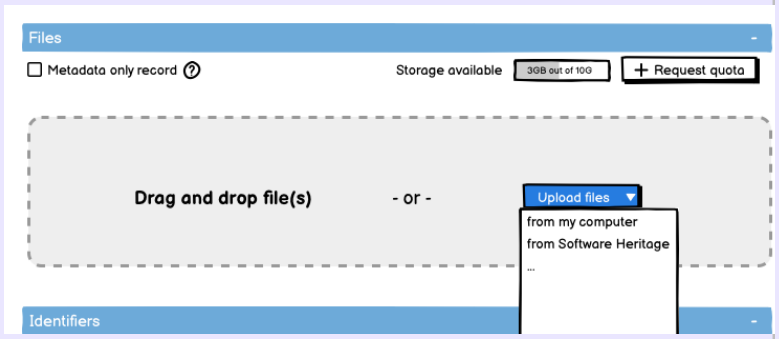
- `datePublished`
- `editor`
- `fileFormat`
- `funder`
- `keywords`
- `license`
- `producer`
- `provider`
- `publisher`
- `sponsor`
- `version`
- `isAccessibleForFree`
- `isPartOf`
- `hasPart`
- `position`
- `description`
- `identifier`
- `name`
- `sameAs`
- `url`
- `relatedLink`
- `givenName`
- `familyName`
- `affiliation`
- `identifier`
- `name`
- `maintainer`
- `embargoDate`
- `funding`
- `referencePublication`
- `readme`

The following Codemeta terms requires new fields:

- `codeRepository`
- `programmingLanguage`
- `runtimePlatform`
- `developmentStatus`
- `operatingSystem`

The following Codemeta terms will not be supported:

Functional specifications			
	- \`targetProduct\` - \`applicationCategory\` - \`applicationSubCategory\` - \`installUrl\` - \`memoryRequirements\` - \`permissions\` - \`processorRequirements\` - \`softwareHelp\` - \`softwareRequirements\` - \`softwareVersion\` - \`storageRequirements\` - \`supportingData\` - \`email\` - \`address\` - \`softwareSuggestions\` - \`contIntegration\` - \`buildInstructions\` - \`issueTracker\` *SWHID* InvenioRDM must also be able to store a Software Heritage Identifier (SWHID). The SWHID will be a fully system managed property that users will not be able to edit. The identifier will be managed using InvenioRDM's internal persistent identifier management system which further enforces that we cannot store duplicate SWHID's.		
4	Import software specific vocabularies Users may describe their record using software-specific metadata (as described above). The following fields will be supported by vocabularies that help drive auto-completion, search filtering and display: - \`programmingLanguage\` - TODO (identify suitable vocabulary) - \`runtimePlatform\` - TODO (identify suitable vocabulary) - \`developmentStatus\` - repostatus.org - \`operatingSystem\` - TODO (identify suitable vocabulary)	M	
5	Software metadata on record landing page The record landing page for a repository record should expose:	H	

Functional specifications			
	- The Software Heritage Identifier (SWHID) in a similar way to how a DOI is exposed. *- Software specific metadata.*		
6	*[Deposit form software fields]()* *The deposit form (i.e. form used for depositing records and/or updating metadata) will add support for the new fields defined above. The form will group software-specific fields.*	*H*	
7	*[Deposit form upload SWH ID]()* *The "Files"-section of the deposit form will add support for uploading/registering files from multiple sources include "from Software Heritage".* *Choosing "from Software Heritage", will enable users to provide a SWH ID of software source code already registered in SWH. Also, this feature can detect if a SWH is already deposited in the repository.*  Files Metadata only record Storage available 3GB out of 10G Request quota Drag and drop file(s) - or - Upload files from my computer from Software Heritage ... Identifiers *![](https://hedgedoc.softwareheritage.org/uploads/ae987491-c321-42dc-94ce-b6b122b42d61.png)*	*L*	
8	*[Metadata formats (import/export)]()* *The following metadata formats will be adapted to support software-specific metadata according to the supported terms:* *- BibTeX* *- CSL (Citation Style Language)* *- CFF (Citation File Format)* *- Codemeta* *- DataCite* *The formats will be supported as:* *- Export option on record landing pages* *- REST API metadata formats*	*M*	

Functional specifications			
9	*[Automatic extraction of metadata]* *The automatic extraction of metadata from software source code will happen in two cases:* *- via the GitHub integration* *- via the depositing of already SWH-registered software* *In both cases, the automatic metadata extraction services to prepopulate the deposit form and possibly automatically publish the record.* *In both cases metadata from Codemeta or CFF will be supported as the source of the metadata being imported.*	*L*	
10	***Deposit from repository to universal source code archive*** *We deposit in SWH published records that must fulfill all of the following conditions:* *- Resource types tagged as software source code* *- Public accessible metadata and files* *- Record must have one or more associated files* *- Max size of all files does not exceed 100 Mib (SWH deposit limit)* *A new version of a record in InvenioRDM is technically a new record, and thus must comply with above conditions.* *Metadata updates to an existing record is allowed after the record is published.* **Event triggers** *The moments when software must be pushed to SWH is when:* *- A draft record is published.* *- A record metadata update is published.* *- An embargo period expires.* *Multiple scenarios exist where a user changes resource type, access rights or embargo period.*	*H*	

Functional specifications			
	These scenarios are explicitly excluded from the scope (see 2.4) *Background publishing to SWH* A record is published to SWH when one of the above events are triggered and the record fulfill all conditions. The publishing to SWH uses the [SWH SWORD deposit API](https://docs.softwareheritage.org/devel/swh-deposit/api/index.html) to send a compressed file and metadata file. *Files* The SWORD deposit requires a single compressed file to be sent. In case a record has a single compressed file (zip, tar, tar.gz, tar.bz2 or tar.lzma) this file will be sent directly. In case a record has multiple files, they will be compressed and sent as a single file. *State management* Internally, InvenioRDM will keep track of records sent to SWH and their submission state. The following state are defined (similar to SWH's [deposit statuses](https://docs.softwareheritage.org/devel/swh-deposit/api/user-manual.html#push-a-deposit)) - Created - Event triggered and record matches - In progress - Sending started (SWH partial) - Sent - Record was sent to SWH (SWH deposited, verified and loading states) - Done - Record succeeded ingestion in SWH (SWH done state) - Failed - Record failed ingestion in SWH (SWH rejected, expired or failed states)		
	Feature flag, configuration and documentation *Configuration* The Software Heritage integration in InvenioRDM will need to be an optional add-on as not all InvenioRDM instances may want to enable the integration, nor is an InvenioRDM instance even guaranteed to host software source code. Software Heritage further requires authenticated access, thus as minimal a		

Functional specifications

	system administrator needs to provide their SWH credentials to the InvenioRDM application. **Documentation** *We will ship InvenioRDM with Software Heritage support directly integrated, but users will need to enable the integration similar to how users need to enable the [DataCite integration](https://inveniordm.docs.cern.ch/customize/dois/) for registering DOIs.*		

- Priority: H=High, M=Medium, L=Low

External references

External references

#	*Short description*	*Reference*
1	*InvenioRDM Infrastructure Architecture*	*https://inveniordm.docs.cern.ch/develop/architecture/infrastructure/*
2	*InvenioRDM Records Data Model*	*https://inveniordm.docs.cern.ch/develop/architecture/records/*

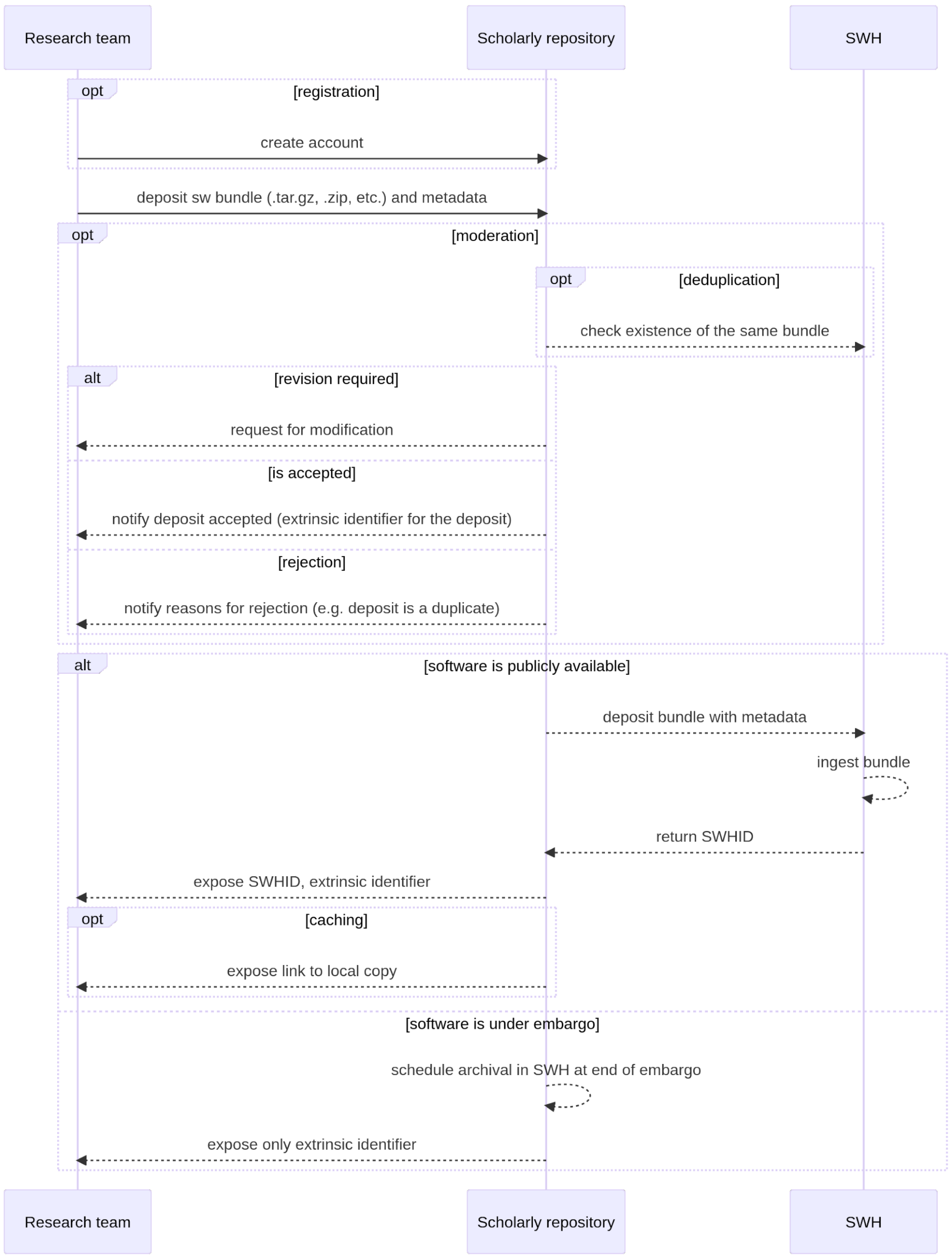
External references - SWH

#	*Short description*	*Reference*
1	*SWORDv2*	*http://swordapp.github.io/SWORDv2-Profile/SWORDProfile.html#protocoloperations_creatingresource_entry*
…		

Annexe

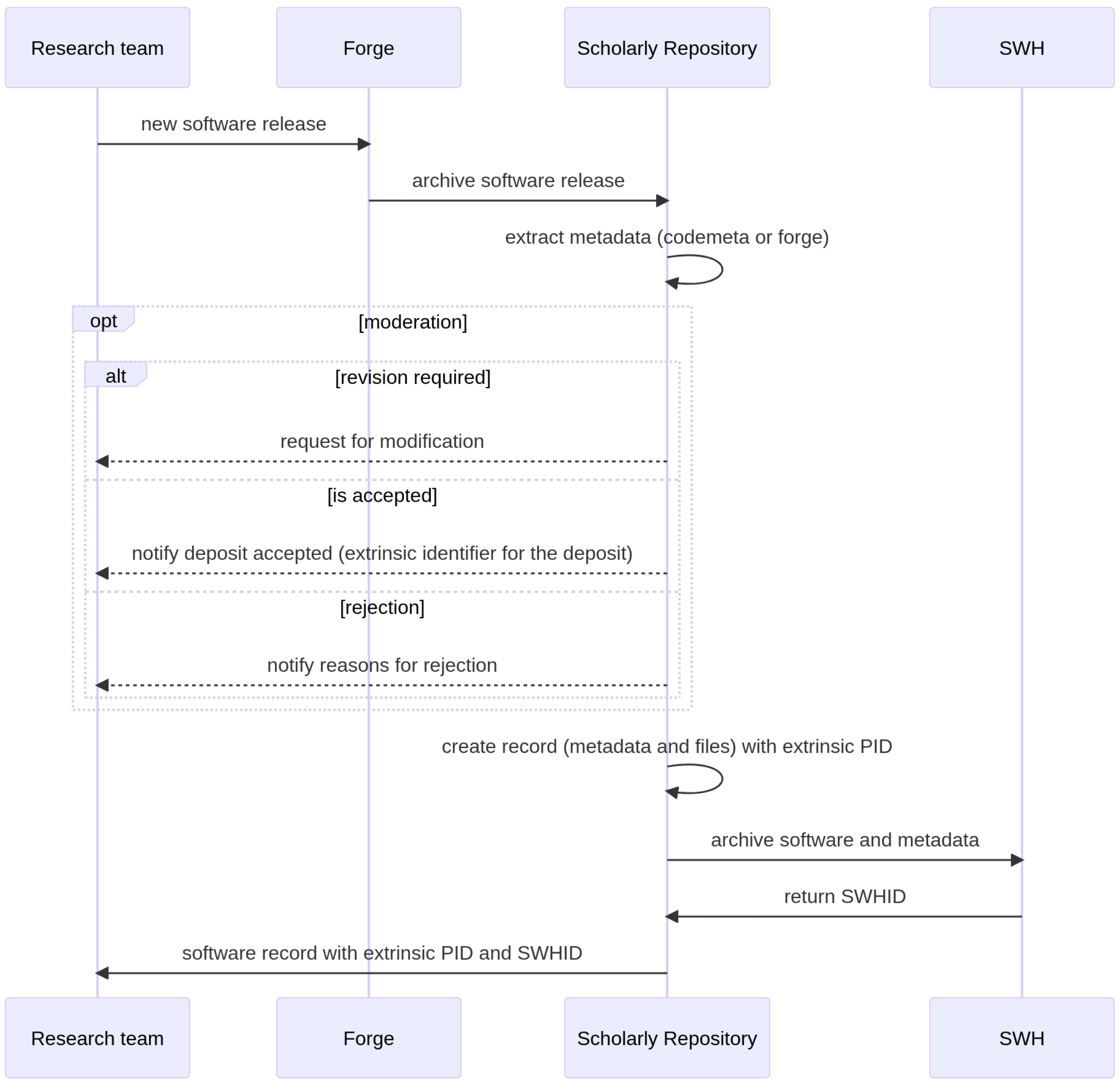
Annexe A: Sequence diagram UC1

`Deposit Bundle Through an InvenioRDM repository`


Research team
Scholarly repository
SWH
opt
[registration]
create account
deposit sw bundle (.tar.gz, .zip, etc.) and metadata
opt
[moderation]
opt
[deduplication]
check existence of the same bundle
alt
[revision required]
request for modification
[is accepted]
notify deposit accepted (extrinsic identifier for the deposit)
[rejection]
notify reasons for rejection (e.g. deposit is a duplicate)
alt
[software is publicly available]
deposit bundle with metadata
ingest bundle
return SWHID
expose SWHID, extrinsic identifier
opt
[caching]
expose link to local copy
[software is under embargo]
schedule archival in SWH at end of embargo
expose only extrinsic identifier
Research team
Scholarly repository
SWH

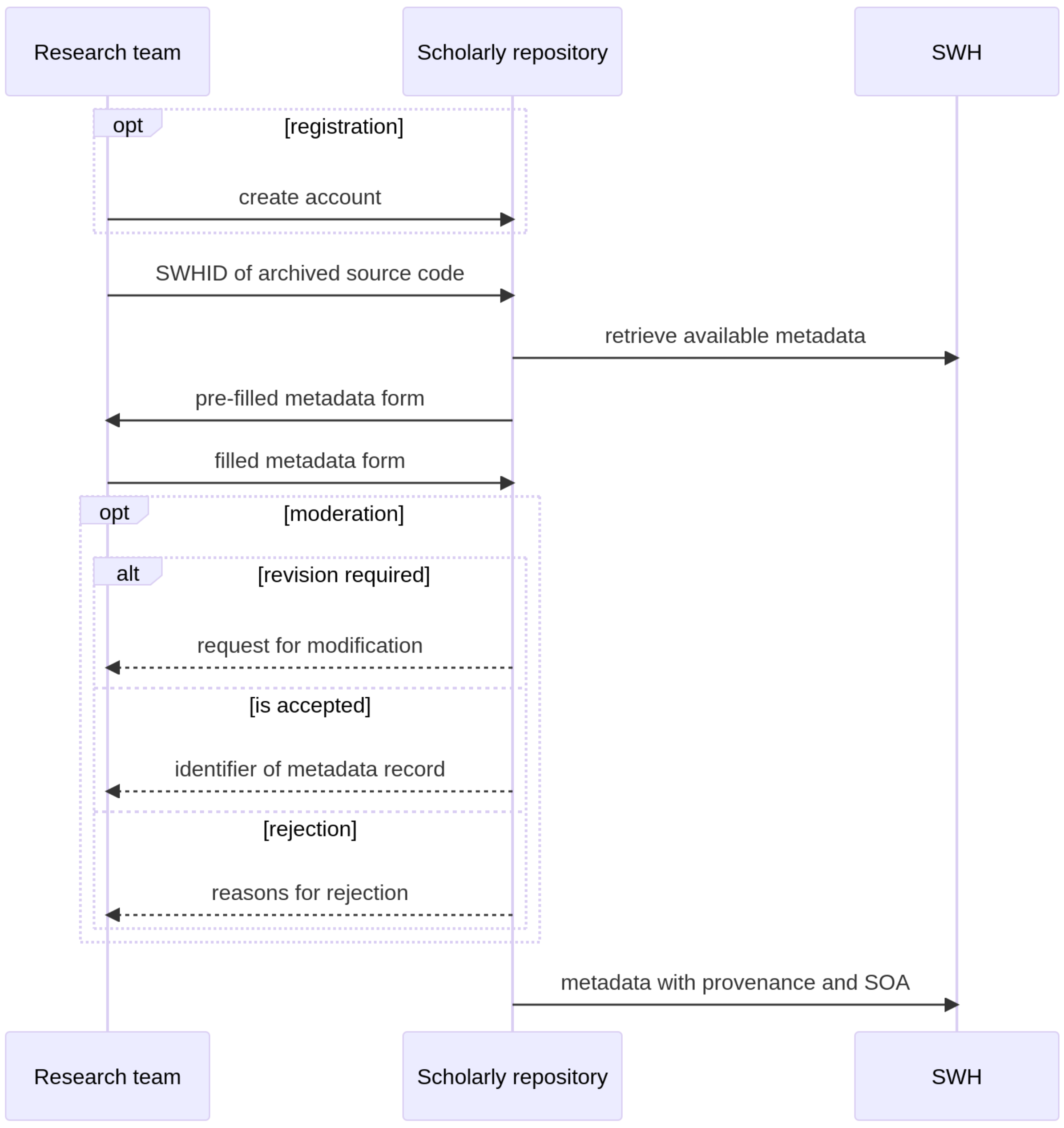
Annexe B: Sequence diagrams UC2

`Automated deposit of New Releases into InvenioRDM and SWH (Manual and Automated)`


Research team
Forge
Scholarly Repository
SWH
new software release
archive software release
extract metadata (codemeta or forge)
opt
[moderation]
alt
[revision required]
request for modification
[is accepted]
notify deposit accepted (extrinsic identifier for the deposit)
[rejection]
notify reasons for rejection
create record (metadata and files) with extrinsic PID
archive software and metadata
return SWHID
software record with extrinsic PID and SWHID
Research team
Forge
Scholarly Repository
SWH

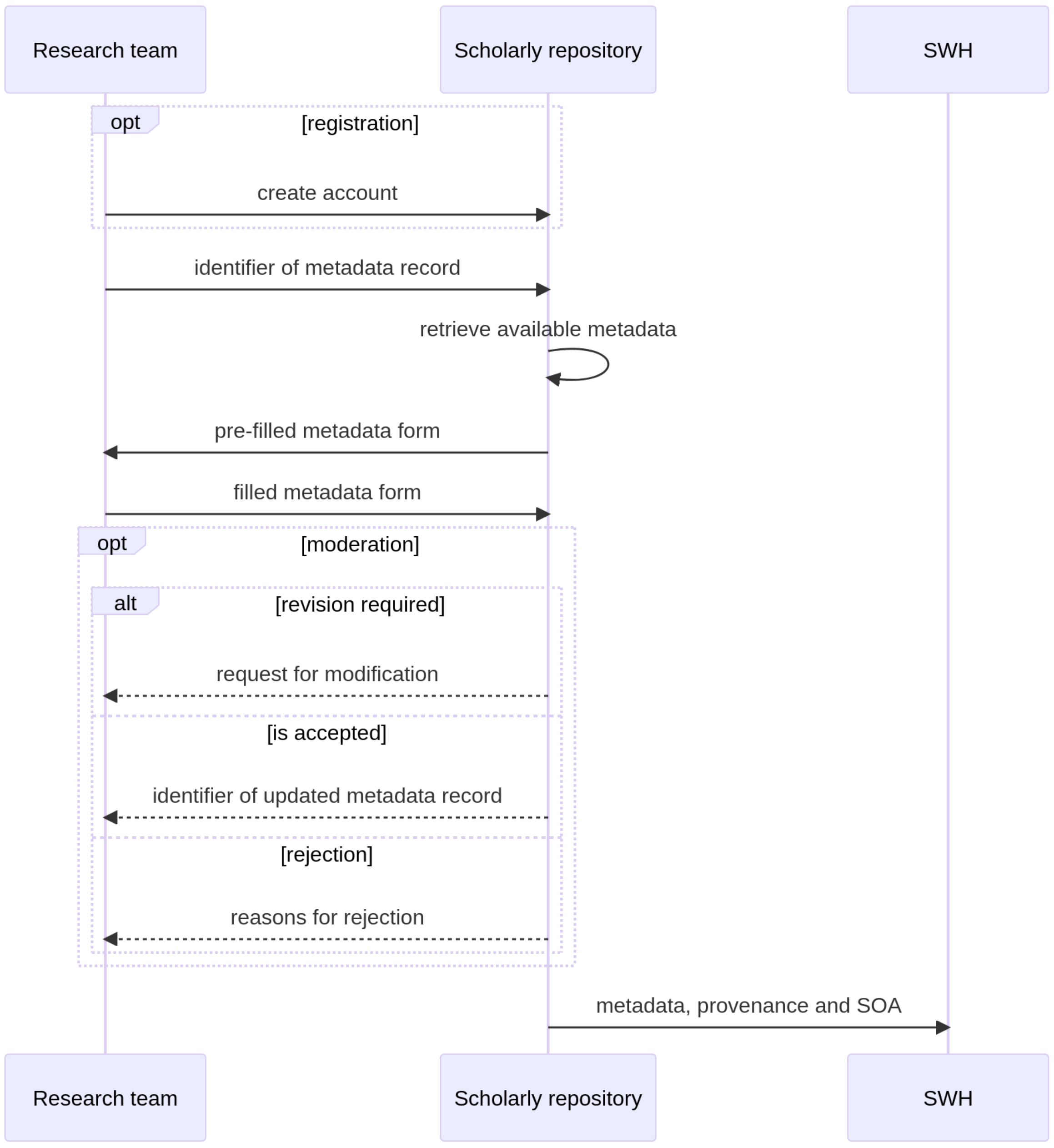
Annexe C: Sequence diagrams UC3

UC3 Registering Already Archived Software in an InvenioRDM repository



Annexe D: Sequence diagrams UC4

UC4 Updating Existing Metadata in an InvenioRDM repository


Research team
Scholarly repository
SWH
opt
[registration]
create account
identifier of metadata record
retrieve available metadata
pre-filled metadata form
filled metadata form
opt
[moderation]
alt
[revision required]
request for modification
[is accepted]
identifier of updated metadata record
[rejection]
reasons for rejection
metadata, provenance and SOA
Research team
Scholarly repository
SWH