

Integrating Persistent Identifiers into repository workflows

Nick Nicholas
Business Analyst, PILIN project

<http://www.pilin.net.au>

Summary for the Impatient

- Persistent Identifiers must be integrated into all repository workflows...
- but integration out of the box is too tight to be adaptable...
- so need reconfiguration or workarounds to adapt them.

Persistent Identifiers

- Objects *identified* independent of location
- Objects *managed* independent of location
- Objects *accessible* independent of location
- Objects managed instead of *instances*
- Long-term *reliability* of access & management
- Strategies: PIDs, Handles, PURLs, (URIs)

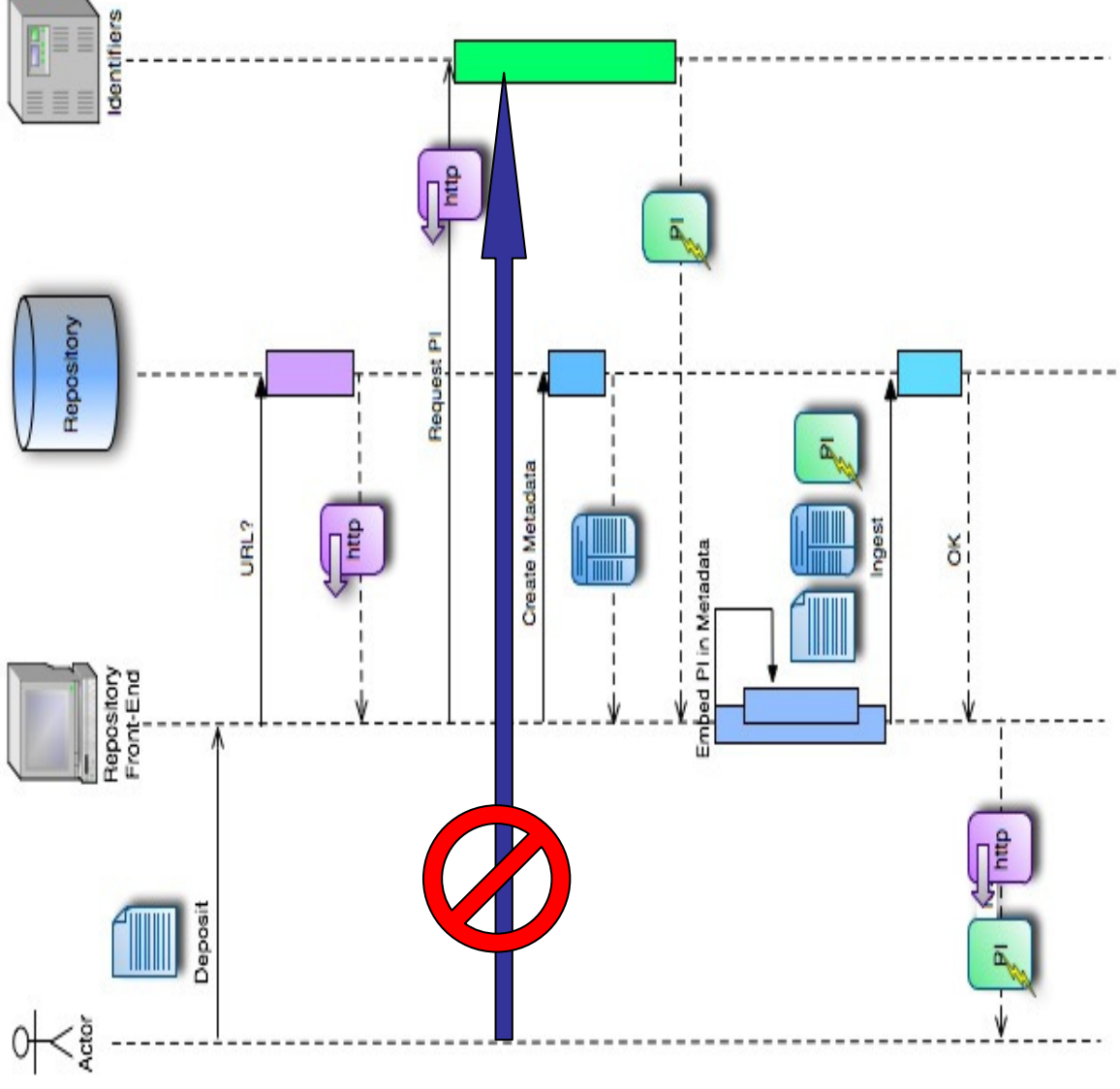
Persistent ID Workflows

- Persistent Identifiers (PIs) impossible to “patch” (= change name) after release
- In a perfect world...
 - All repository workflows should use PIs
 - To guarantee their location-independence
 - Objects branded with PIs
 - To better survive move outside repository
- So, Objects assigned PIs early in lifecycle

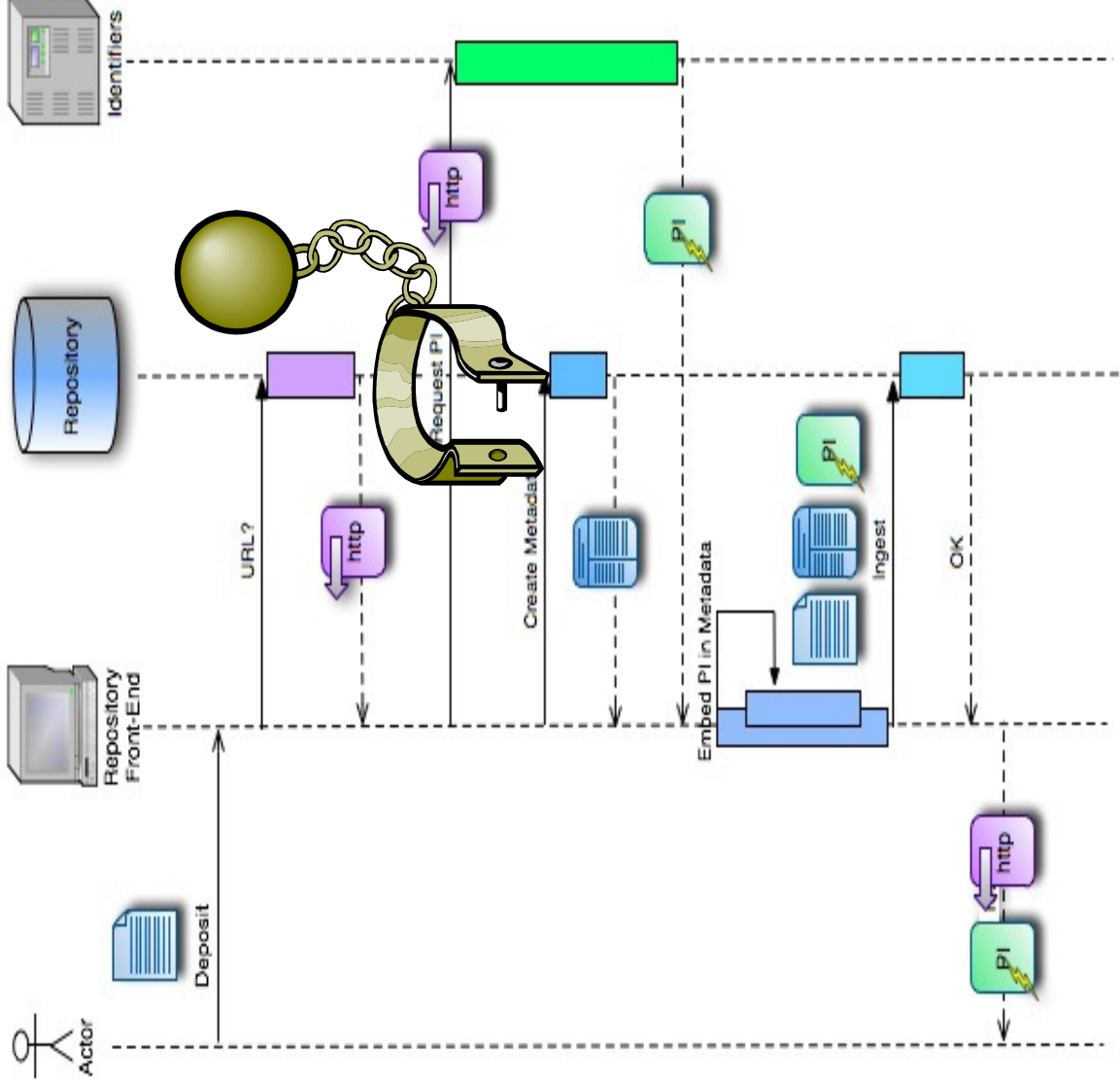
PI Integration into Repositories

- Repositories use PIs in workflows
- PIs most sensible for managed data
- So PIs are repository managers' concern
- PIs become repository managers' responsibility
 - Repository managers make sure PIs are there
 - But shouldn't sideline external PIs (difficult!)
- PIs pervasive in workflows
 - so tight integration

Typical Ingest Workflow



Tight Integration Issues



- PI creation locked in to Ingest process
- Guarantee PI is created when it's needed
- and when it's not needed...

Tight Integration Problems

- Re-ingest object = Re-assign PI
 - Check out object and modify it
 - Reupload the object...
 - New identifier!
- Already assigned PI ignored
 - Pre-automation workflows
 - Externally packaged content (with e.g. DOIs)
 - Ingested object is only appropriate copy:
PI shared among all copies in federation

Tight Integration Problems

- No management of PIs outside repository:
 - Locked in to information model of repository
 - Related objects?
 - Appropriate copies of object in federation?
 - Different resolution protocols?
 - Aggregations/Disaggregations?
- No embedding PIs before ingest
 - Want to write my own metadata with PI
 - Or watermark PI into object

Tight Integration Problems

- PI solution specific to Repository
 - Migrating from Fedora to DSpace?
- So: PI Coupling too tight
- No Repository manager override on PIs
- Things made *too* simple for repository managers

PILIN Project



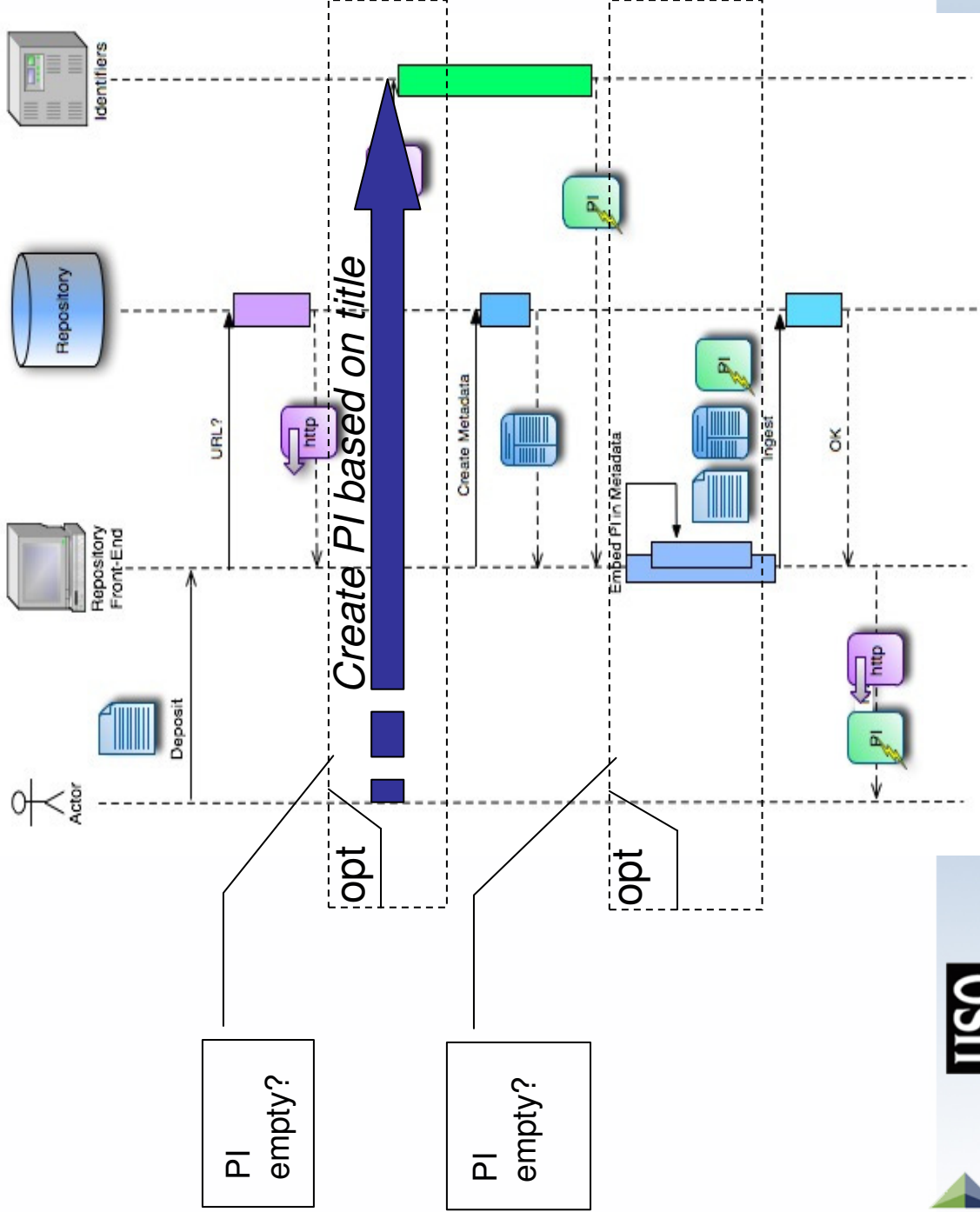
- <http://pilin.net.au>
- Strengthening Australia's ability to use global identifier infrastructure
- e-Learning, e-Research, e-Library
 - i.e. Repositories, Repositories, Repositories
- Developing: identifier toolkits
 - Using Handle as exemplar technology:
popular with repositories
- Developing sample applications
- Developing policy guidance

<http://www.pilin.net.au>

Service-oriented design

- If coupling is tight... loosen it.
- PIs managed through uncoupled services
 - Define services from outside repository to manage PIs
- Given services, can arrange own workflow
- PILIN: JAHDL (Java API for Handle)
- Future: Web services for Handle
 - Agnostic about repository type
 - (Ultimately) agnostic about PI scheme

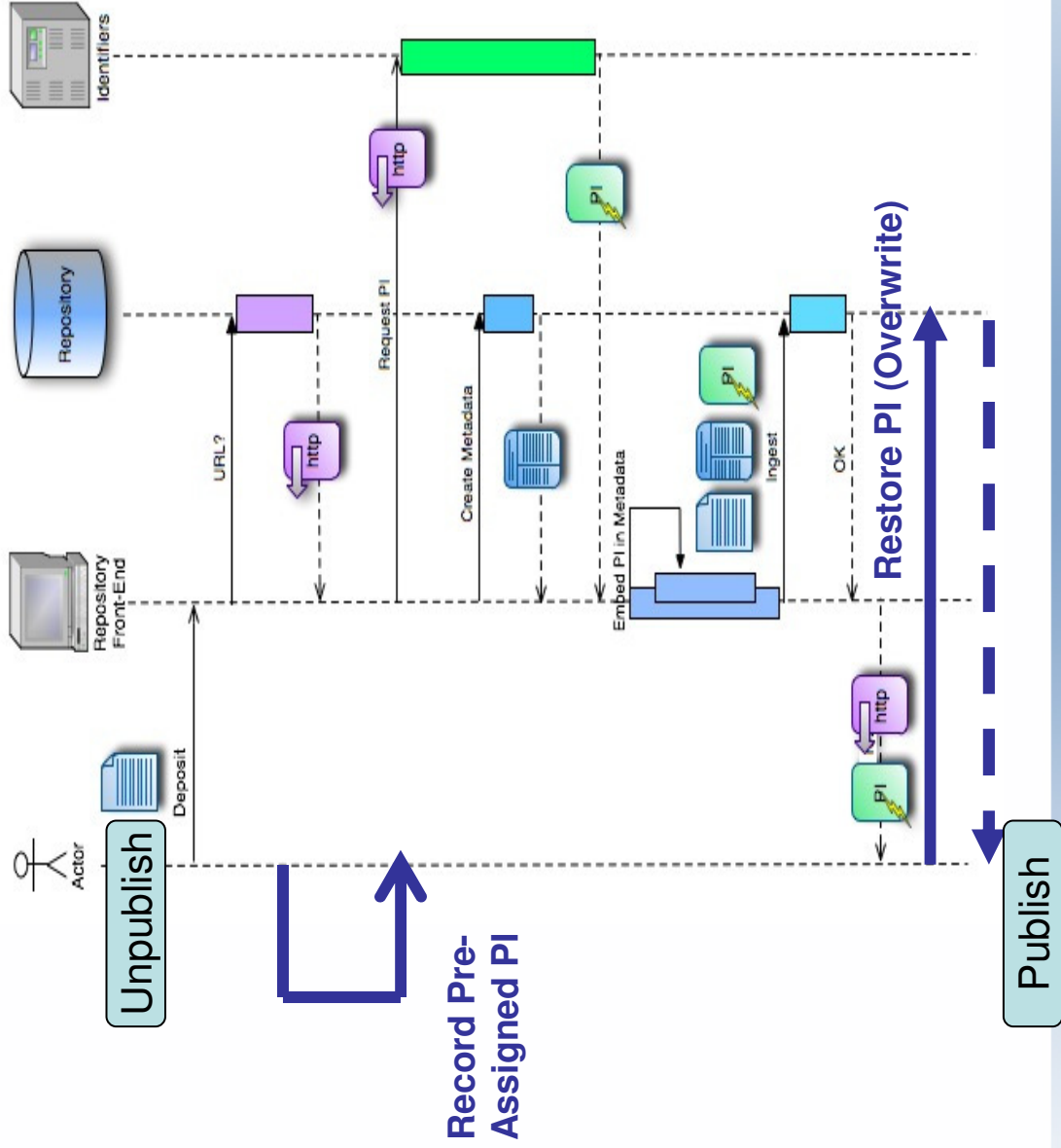
Create with my PI format, unless PI already there



Manual override of PIs

- Often hard for repository managers to break into repository automation
 - Lack of programming chops
 - Or repository system closed off
- Alt: allow managers manual access to PI system
- i.e. Allow managers to “patch” PIs
 - OK, so long as done before publishing PIs
- PILIN Web Handle Management Tool
 - Includes Batch mode

Patch unwelcome new PI



<http://www.pilin.net.au>

Patch unwelcome new PI

- But if PI is patched, what stays constant?
 - ... filename
 - Can also brand proper PI into object
- ARROW instance of workflow:
 - Filename (based on local ID scheme):
constant
 - Retrieve old PI from Handle logs
 - Retrieve new PI from Ingest logs
 - Use batch tool to patch PIs

Conclusion

- Turnkey integration is safe option
 - We use DSpace and VITAL instead of Fedora API calls
- But does not anticipate all use cases or configurations
 - Shouldn't have to reingest to fix typos, true
 - But will reingest when migrating content

Summary

- Pls are needed in repositories
- Pls need to pervade repository workflows
- Pl integration out of the box: not adaptable
- To make Pl integration adaptable:
- Take over workflows: uncoupled services
 - Workflow engine: The “right solution”
- Or patch Pls as needed before publishing
 - Manual override: The “close enough solution”

Summary

- Manual override should not mean open slather
 - Need good governance
 - (as do all PI systems) (and repositories)
- Must get workflows right: persistent is a long time
- Questions?