Proven Documentation

Release 2.0

Bibi Mathew, Eric Stephan, Todd Elsethagen

Feb 19, 2021

Contents

1 Overview **1**

2 Background **3**

3 Activities **5**
- 3.1 Provenance . . . 5
- 3.2 Streaming . . . 5
- 3.3 Hybrid Store . . . 5

4 Users Guide **7**
- 4.1 Scientific Workflow Reproducibility . . . 7
- 4.2 Power Grid Simulation Data Management . . . 7

5 Installing Proven **9**
- 5.1 Purpose . . . 9
- 5.2 What you should expect to do . . . 9
- 5.3 Prerequisites . . . 9
- 5.4 Clone Proven Repositories . . . 10
- 5.5 Import Gradle Projects in Eclipse . . . 10
- 5.6 Create General Eclipse Project for testbed Resources . . . 10
- 5.7 Build and publish proven_message jar . . . 11
- 5.8 Build and publish proven_message-0.1-all-in-one jar . . . 13
- 5.9 Building the ProvEn Server (proven-member) . . . 13
- 5.10 Create External Tools Configurations . . . 15
- 5.11 Create Debug Configuration . . . 15
- 5.12 Running the Hybrid Service . . . 15
- 5.13 Swagger UI of Debug Interface . . . 17

6 Proven Componenets **19**
- 6.1 Inside Proven Member . . . 19
- 6.2 Proven Exchange . . . 19
- 6.3 Proven Hybrid Store . . . 28

7 Research Areas **29**

8 License **31**

9 Indices and tables **33**

CHAPTER 1

Overview

Proven is a hybrid data platform (HPD) that supports modeling and simulation (M&S) studies and workflow reproducibility by combining an off the shelf (OSS) time-series database, a triple store, and real-time streaming technologies. Proven can be currently used by InfluxDB, relies on an internal Sesame triple store, and uses the Hazelcast In-Memory-Data-Grid (IMDG).

The initial version 1.0 release of Proven used to support reproducibility can be found on the ProvenanceEnvironnent main website. The GridApps-D project uses proven-docker as a means to deploy Proven as part of its application developer system. The current proven development is located on proven-cluster, proven-client, and proven-docker.

CHAPTER 2

Background

CHAPTER 3

Activities

3.1 Provenance

3.2 Streaming

3.3 Hybrid Store

CHAPTER 4

Users Guide

4.1 Scientific Workflow Reproducibility

4.2 Power Grid Simulation Data Management

CHAPTER 5

Installing Proven

5.1 Purpose

- Setting up a development and testbed environment is not trivial. This slide deck documents the testbed I set up on my MacOS laptop. Hopefully this will be helpful to the wider GridAPPS-D team or other development teams using ProvEn.
- Disclaimer: This guide is intended to offer a complete set of notes. However there may be differences depending on the platform you are using and unfortunately there may some gaps of knowledge.

5.2 What you should expect to do

Once the development system and testbed are completely setup you should be able to run a ProvEn server in debug mode, accessible by REST services

5.3 Prerequisites

- **Download and install**
  - Latest Eclipse IDE J2EE (I used Eclipse Oxygen.2 (4.7.2))
  - Java 8 JDK
- **Brew install**
  - git 2.12.0
  - gradle 4.5.1
  - influxdb 1.4.2
  - maven 3.3.3 3.3.9

- **Download and set aside for later use**
  - payara-micro-5.181.jar from: https://s3-eu-west-1.amazonaws.com/payara.fish/Payara+Downloads/
- Please note that Eclipse will need to be configured to support your Gradle, Maven, use your Java 8 JDK

5.4 Clone Proven Repositories

- https://github.com/pnnl/proven-message
- https://github.com/pnnl/proven-cluster
- https://github.com/pnnl/proven-client
- https://github.com/pnnl/proven-docker

5.5 Import Gradle Projects in Eclipse

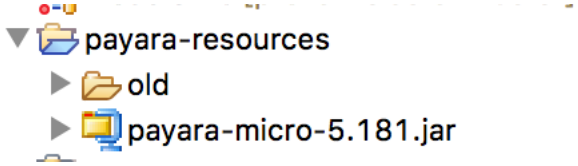
- **Import proven-message and proven-member projects as gradle projects.**
  - Note: The "proven-cluster" project contains several nested layers of projects.
  - Import the "proven-cluster" subproject "proven-member" – importing "proven-cluster will cause undesirable effects, limiting what you can build.

5.6 Creat Gen- eral Eclipse Project for testbed Re- sources

This project(name it "payara- resources") will be used to pro- vide a

micro service engine

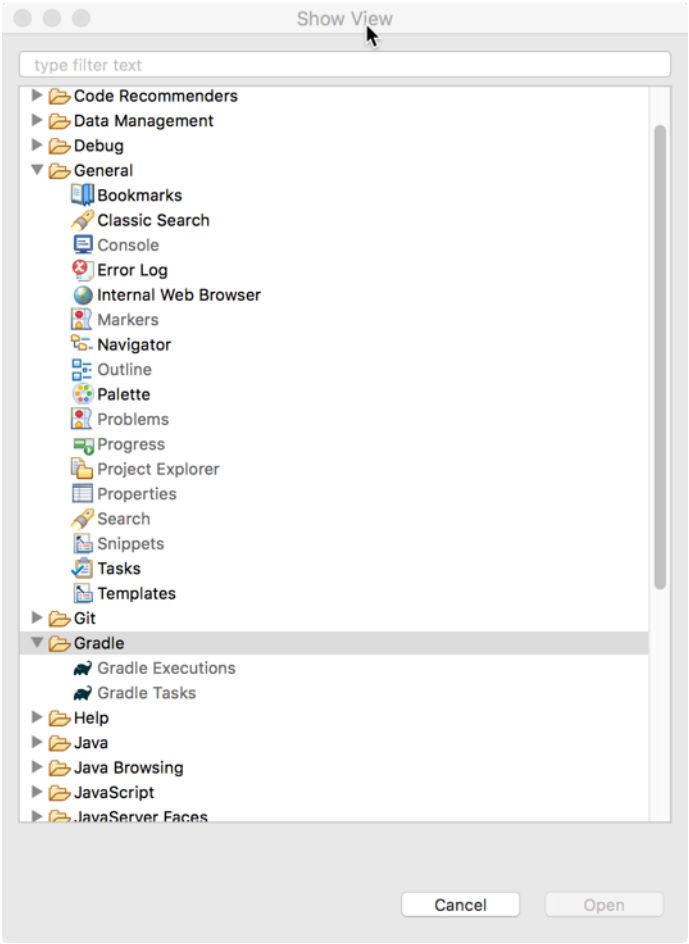
for testing later. Add the payara-micro jar in the top folder



5.7 Build and publish proven_m jar

- **Open the following**
  - General->Console
  - Gradle->Gradle Executions
  - Gradle->Gradle Tasks
- Click on the proven_messag project (you may need to

click on the build.gradle file).



5.8 Build and publish proven_m 0.1-all-in-one jar

Build and publish the proven_messag 0.1-all-in-one.jar file to maven local repository

so that the hybrid services can use the interface. * Open build task folder * Double click on “build” task. * Open publishing task folder. * Double click on “publish” task. * Double click on “publishToMavenLocal” * Confirm no errors in Console View. * Inspect the proven-message/build/libs/ directory for proven-message-0.1-all-in-one.jar

5.9 Building the ProvEn Server (proven-member)

- Use Gradle Tasks

to Build the Proven hybrid service war file

- If necessary use Gradle IDE tasks to rebuild eclipse files.

5.10 Create External Tools Configurations

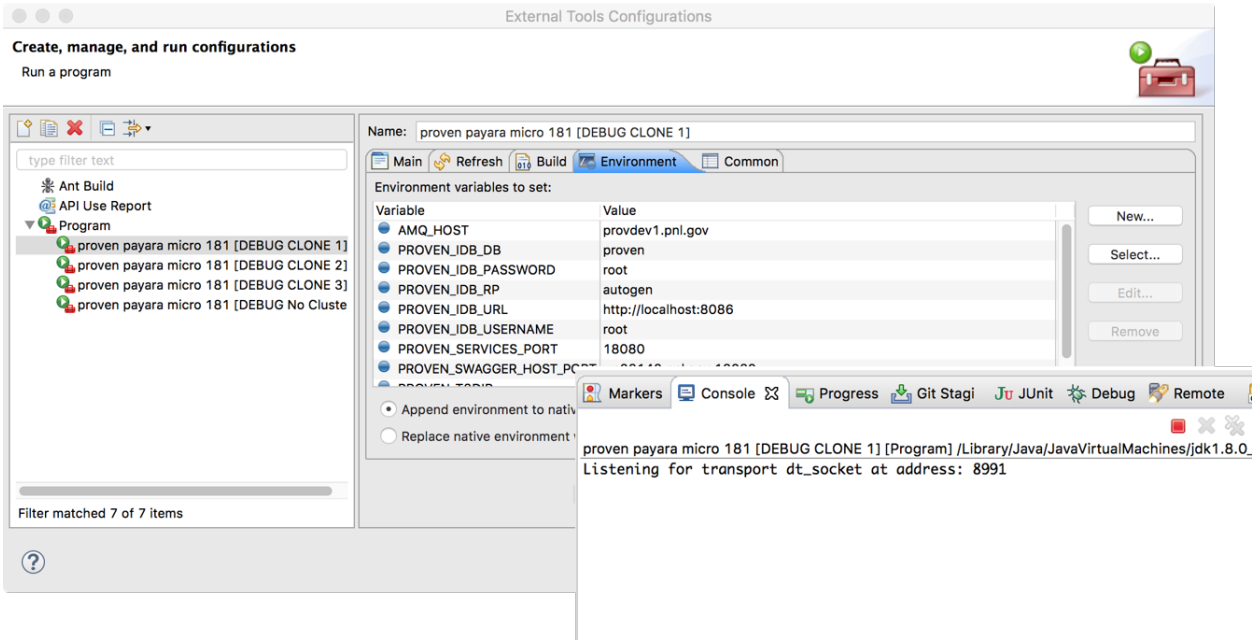
5.11 Create Debug Configuration

5.12 Run the Hybrid Service

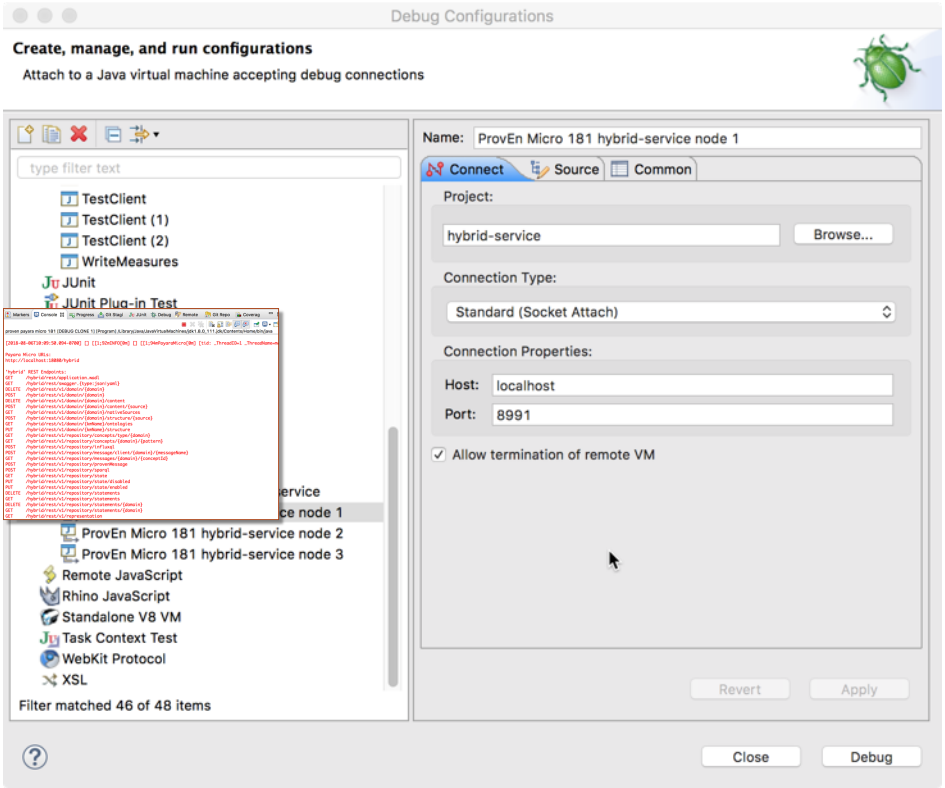
- **Steps to running ser**
  - Start InfluxDB
  - Run External Tools Con-

figurations "proven payara micro 181 [DEBUG CLONE 1]"

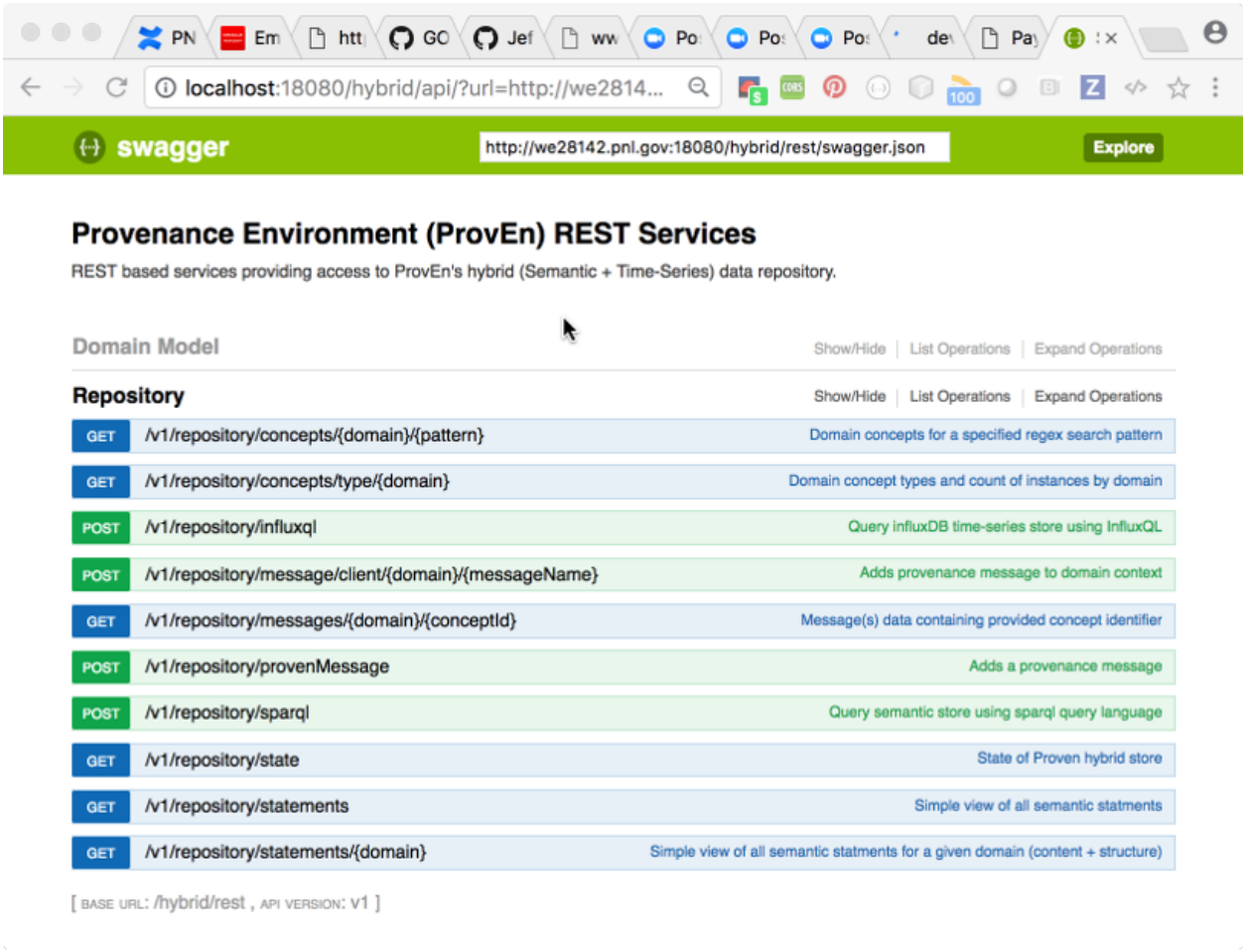
- Run debug configuration "proven micro 181 hybrid-service node 1"
- Startup can take several minutes



Correct startup should look something like this in the console



5.13 Swagger UI of Debug Interface


localhost:18080/hybrid/api/?url=http://we2814...
swagger
http://we28142.pnl.gov:18080/hybrid/rest/swagger.json
Explore
Provenance Environment (ProvEn) REST Services
REST based services providing access to ProvEn's hybrid (Semantic + Time-Series) data repository.
Domain Model
Show/Hide | List Operations | Expand Operations
Repository
Show/Hide | List Operations | Expand Operations
GET /v1/repository/concepts/{domain}/{pattern} Domain concepts for a specified regex search pattern
GET /v1/repository/concepts/type/{domain} Domain concept types and count of instances by domain
POST /v1/repository/influxql Query influxDB time-series store using InfluxQL
POST /v1/repository/message/client/{domain}/{messageName} Adds provenance message to domain context
GET /v1/repository/messages/{domain}/{conceptId} Message(s) data containing provided concept identifier
POST /v1/repository/provenMessage Adds a provenance message
POST /v1/repository/sparql Query semantic store using sparql query language
GET /v1/repository/state State of Proven hybrid store
GET /v1/repository/statements Simple view of all semantic statments
GET /v1/repository/statements/{domain} Simple view of all semantic statments for a given domain (content + structure)
[BASE URL: /hybrid/rest , API VERSION: v1]

CHAPTER 6

Proven Componenets

Proven consists of the following primary architectural elements:

1. Exchange - Data collection, preparation, and distribution to Hybrid Store's streaming environment.
2. Hybrid Store – Message streaming and cache, with archival (TS, T3, Object). Messages are RDF sub-graphs. Stream processing support.

6.1 Inside Proven Member

6.2 Prove Ex- change

- Data col- lec- tion and prepa- ra- tion for dis- tri- bu- tion to

Hybrid Store

- External - JSON and JSON-LD are currently the accepted formats
- Internal – JSON-LD (i.e. data represented as subgraphs internally)

- **All disclosed conten**

1. Proven-message is verified (i.e. wrapper; the proven part)
2.

Message content is verified. If content is JSON-LD then JSON-SCHEMA for JSON-LD is used to verify syntactic correctness.

- **All disclosed conten**

  1. Mime type is examined if JSON-LD then no further processing necessary
  2. For domain Knowledge content: A default JSON-LD

context is provided in the outer object. This simply includes a @vocab setting using the message's domain value.

3. For "Proven specific" content: The predefined context for the message content type is injected.

4. If the message's outer object is an array, then the array is first encapsulated by an object before adding the context.

- All proven specific content types

(including the wrapper) have an associated ontology and context definition.

- Message syntactic verification and semantic transforms are performed at point of entry and any issues are reported to Hybrid store's Response stream.
- **Exchange consists o**
  - Exchange Buffer: Have unique responsibilities in terms of disclosure item processing
  -

Module Exchange: Responsible for distributing a disclosed item to a

“ready” ExchangeBuffer for processing (distribution can be at the module, member or cluster level)

- **Following are the E**
  - DisclosureBuffe
    dis-
    clo-
    sure
    item
    dis-
    tri-
    bu-
    tion
  - ModuleServicel
    ser-
    vices
    mod-
    ule
    re-
    quests.
  - PipelineService
    ser-
    vices
    pipeline
    re-
    quests.
  - ResponseBuffer
    dis-
    tri-
    bu-
    tion

of response/results to domain resonse stream.

- ProvenanceBuffer: provenance generation and distribution
- RulesBuffer: rule-based inference and distribution

• Provenance capture is accomplished using the aforementioned message ontologies and SHACL rules to generate PROV provenance assertions.

- Rules are also defined using SHACL and content specific ontologies.
- Each domain has its own provenance and stream.
- Each buffer is applicable to any domain; processing is determined by a message's semantic description or Message Model (i.e. ontologies, context, rules, provenance, etc.)
- Disclsoure item paths are

static and these paths are defined and used by a Module-

Exchange.

- ModuleExchang
are informed of the candidate ExchangeBuffers via module reporting making their lookup performant.
- Back pressure is to the caller.
- Exchange items that cannot be processed due to

an "unavailable" ModuleExchange, are transferred to a Suspend stream to avoid data loss. These items are given highest priority once a ModuleExchange becomes available.

6.3 Prove Hybrid Store

CHAPTER 7

Research Areas

- Distributed SPARQL query
- Stream reasoning
- Architecture (Kappa)
- Dynamic reference data
- ML/NLP guidance and support
- Standards based SPARQL-Stream query
- Out of order requirements*

CHAPTER 8

License

Battelle Memorial Institute (hereinafter Battelle) hereby grants permission to any person or entity lawfully obtaining a copy of this software and associated documentation files (hereinafter the Software) to redistribute and use the Software in source and binary forms, with or without modification. Such person or entity may use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and may permit others to do so, subject to the following conditions: Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimers. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution. Other than as used herein, neither the name Battelle Memorial Institute or Battelle may be used in any form whatsoever without the express written consent of Battelle.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL BATTELLE OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

General disclaimer for use with OSS licenses

This material was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor the United States Department of Energy, nor Battelle, nor any of their employees, nor any jurisdiction or organization that has cooperated in the development of these materials, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness or any information, apparatus, product, software, or process disclosed, or represents that its use would not infringe privately owned rights.

Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof, or Battelle Memorial Institute. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

CHAPTER 9

Indices and tables

- genindex
- modindex
- search