Analysis Workbench

CyPSA View Catalog

Description	Parameters	Implementation
CyPSA Analysis Workbench allows one to analyze CyPSA-inventoried assets 0. Select a model 1. Select an analysis 2. Show analysis template 3. Populate template with inputs 4. Get analysis output (tabular or graph)	*baseURL:* Base URL for the service that returns tabular data	This view is a composition of a few different views. 0) (GetAnalyses) to provide a dropdown of available analytics 1) (GetAssetInventory, HierarchyEditor) 2) A simple HTML form for GetAssetExposure Analysis. 3) (GetAssetExposure, GraphEditor I TableEditor) Over time, the analysis may change. in which case we want a different template/form for each analysis.

Interface and User Experience

Model Inventory

- 8 Bus
  - + Capital City
  - + Control Center
  - + Cyprus Creek
  - + Haverbrook
  - + North Haverbrook
  - + Ogdenville
  - + Paris
  - + Shelbyville
  - + Springfield

Analysis:

Asset Exposure

Description: Given a network and a set of vulnerability scores, compute the asset exposure.

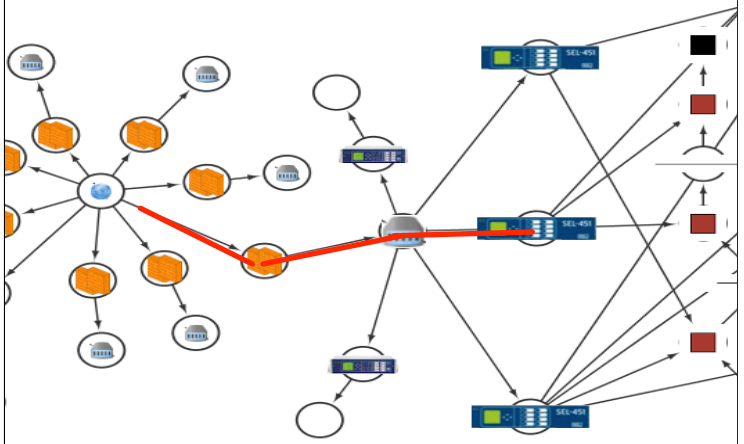
Inputs:
Network: *Ogdenville*
Vulnerability Scores: NVD CVE

Provider: NetAPT Localhost

Output: O Table ⊗ Graph

Run

Asset Exposure Paths

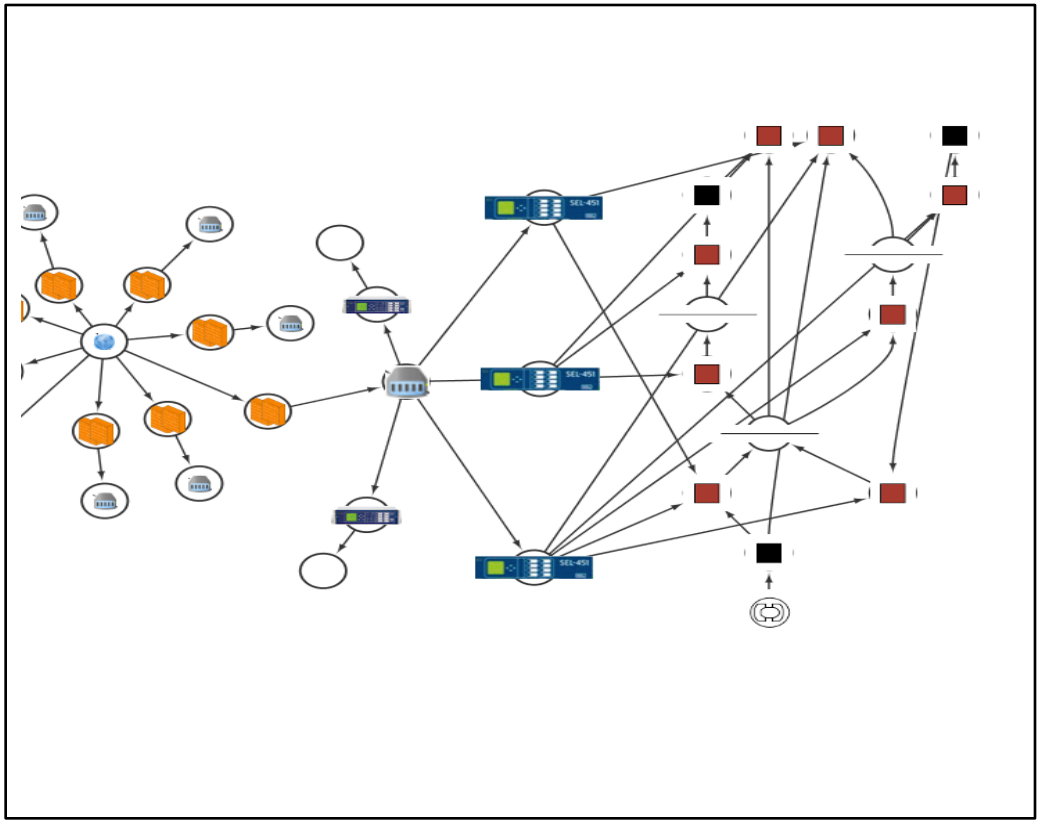


Graph Editor

CyPSA View Catalog

Description	Parameters	Implementation
Graph Editor allows one to browse CyPSA-inventoried assets graphically.	*baseURL:* Base URL for the service that returns a graph to be visualized such as **GetGraph**, or **GraphJoin**	The d3.js library may prove helpful.

Interface and User Experience



Hierarchy Editor

CyPSA View Catalog

Description	Parameters	Implementation
Model Inventory Editor allows one to hierarchically browse and *modify* hierarchical data (e.g. result of a **GetAssetInventory** service call).	*baseURL:* Base URL for the CyPSA Service Endpoint that impls **GetAssetInventory** *config:* name of the project inventory file to use. (optional)	The Yahoo UI library provides a TreeView control that may prove helpful. The view pulls information from a call to the CyPSA **GetAssetInventory** service. We need to restrict what kind of hierarchies can be displayed so as to know how to display them.

Interface and User Experience

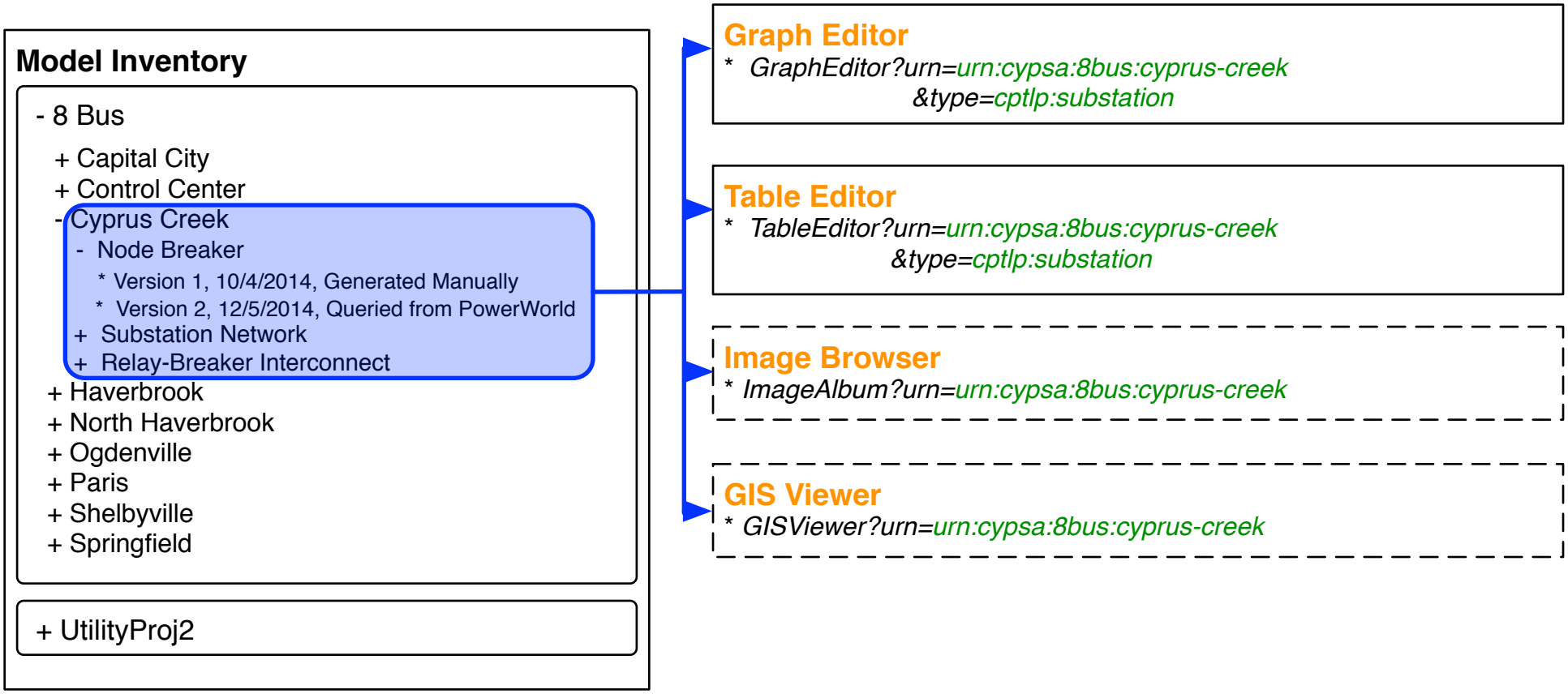
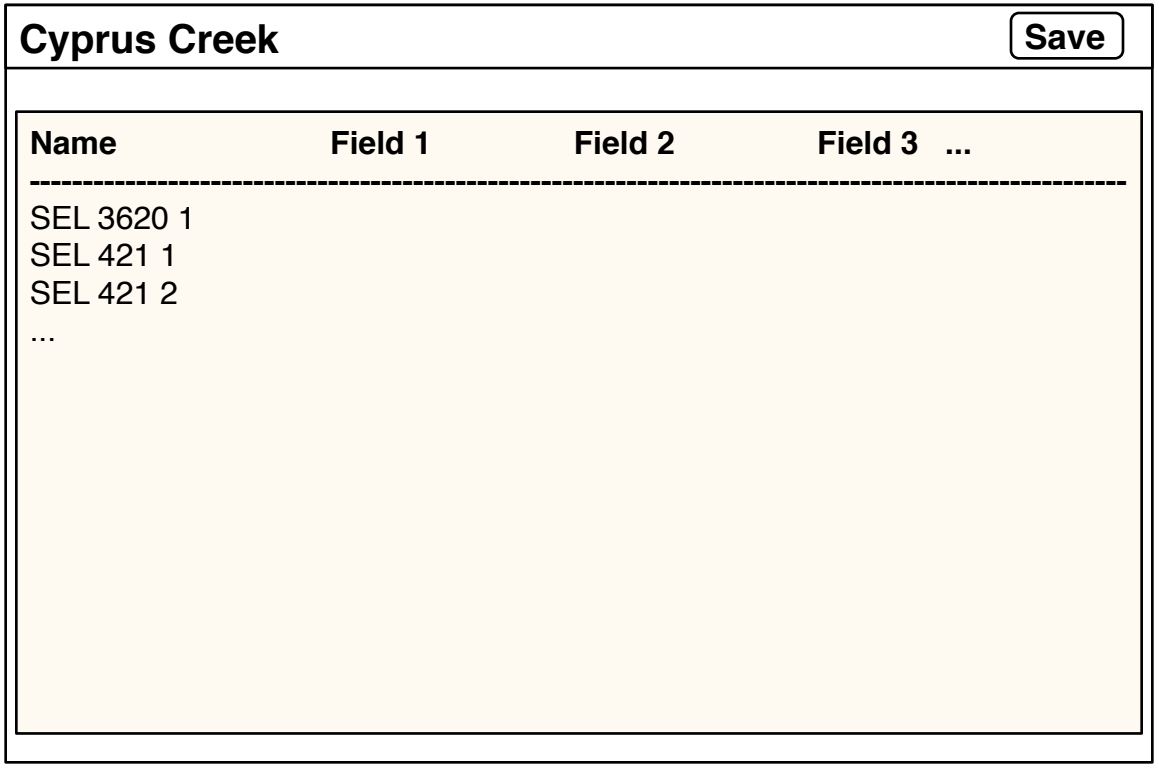


Table Editor

CyPSA View Catalog

Description	Parameters	Implementation
Table Editor allows one to browse CyPSA-inventoried assets or the results of an analysis (e.g. **GetValidReff**, **GetRankedAssets**) via a spreadsheet-style form.	*baseURL:* Base URL for the service that returns tabular data	The Yahoo UI data table may do the trick (http://yui.github.io/yui2/docs/yui_2.9.0_full/datatable/). The view pulls information from a call to a CyPSA service that returns tabular data.

Interface and User Experience



Data Provider Registry

Description	Parameters	Implementation
CyPSA Data Provider Registry allows analysts to register data sources to provide information about CyPSA-inventoried assets and analyses.		

Interface and User Experience

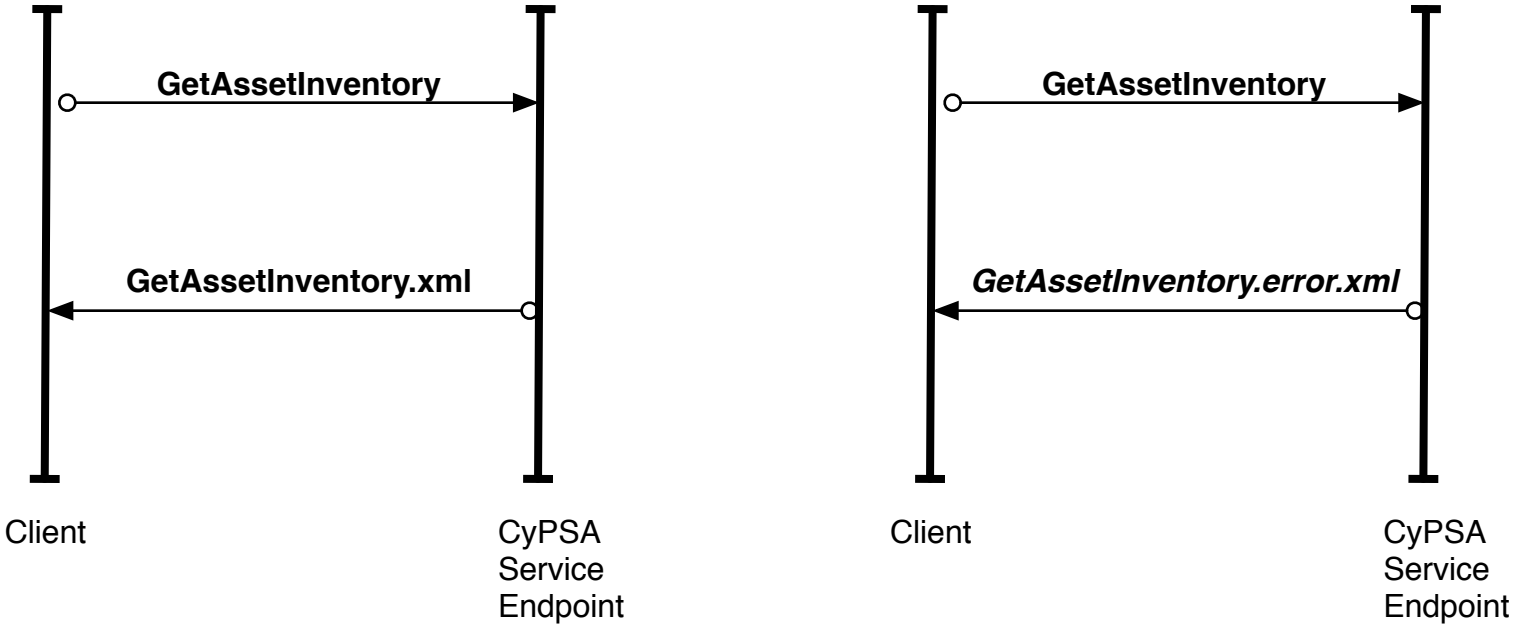
+ IDS Alerts
+ NPView

GetAssetInventory

CyPSA Service Catalog

Description	Parameters	Implementation
GetAssetInventory returns a catalog of metadata for CyPSA-managed projects and the cyber-physical assets contained therein.	No parameters are required. There is an optional *config* parameter, however. *config:* name of the project inventory file to use	The inventory could be contained in a static file or dynamically generated based upon the registered data providers. Response not necessarily in XML (could do JSON too) Test Data: `projects/services/test/GetAssetInventory.xml` We need to write a schema to validate inventory.

Request and Response

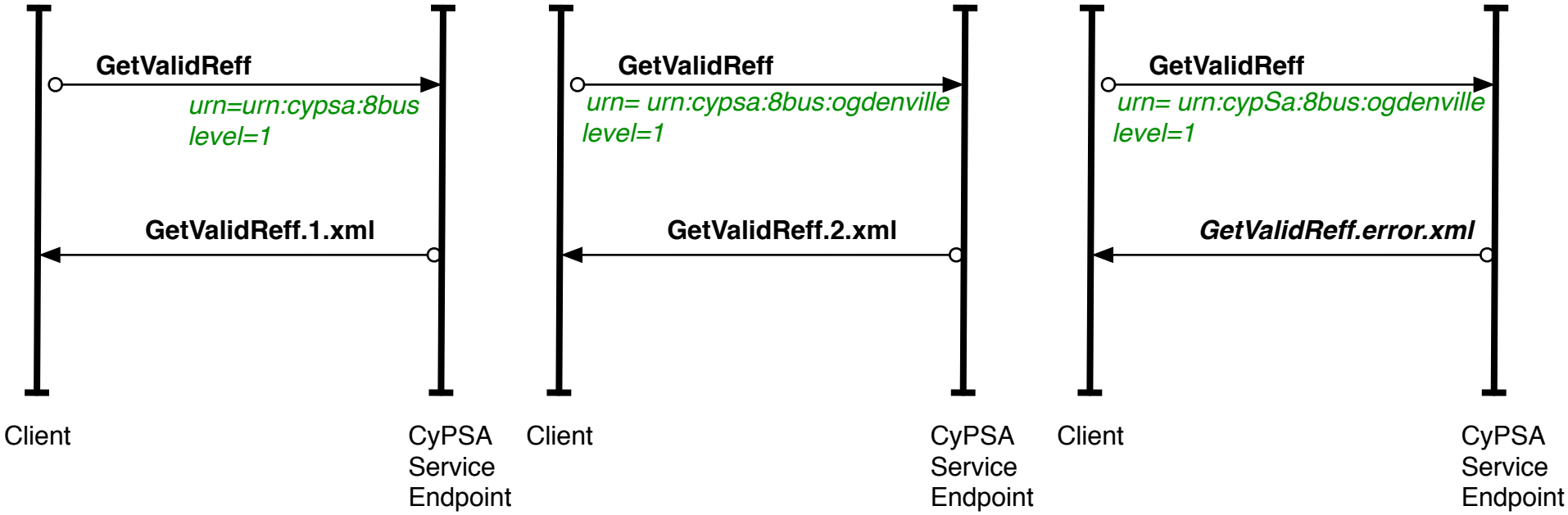


GetValidReff

CyPSA Service Catalog

Description	Parameters	Implementation
GetValidReff returns a list of valid references contained within a given reference. This is useful for discovering unknown assets contained within a known reference.	No parameters are required. There is an optional *config* parameter, however. *urn:* The CyPSA URN *format*: Default is XML, JSON? *level:* Optional, specifies how deep to traverse the reference tree. (1 gives you child references from the request urn, 2 gives you grandchild refs, by default level is set to infinity to give total tree depth)	This could be accomplished in a variety of ways. I'm curious about a DNS-like service, however. Note there is a tradeoff when exploring the reference space for CyPSA assets. You can get by with fewer requests to the service if the *level* parameter is set higher. However, the number of references returned by such a request is higher. Should be able to validate the output of this

Request and Response

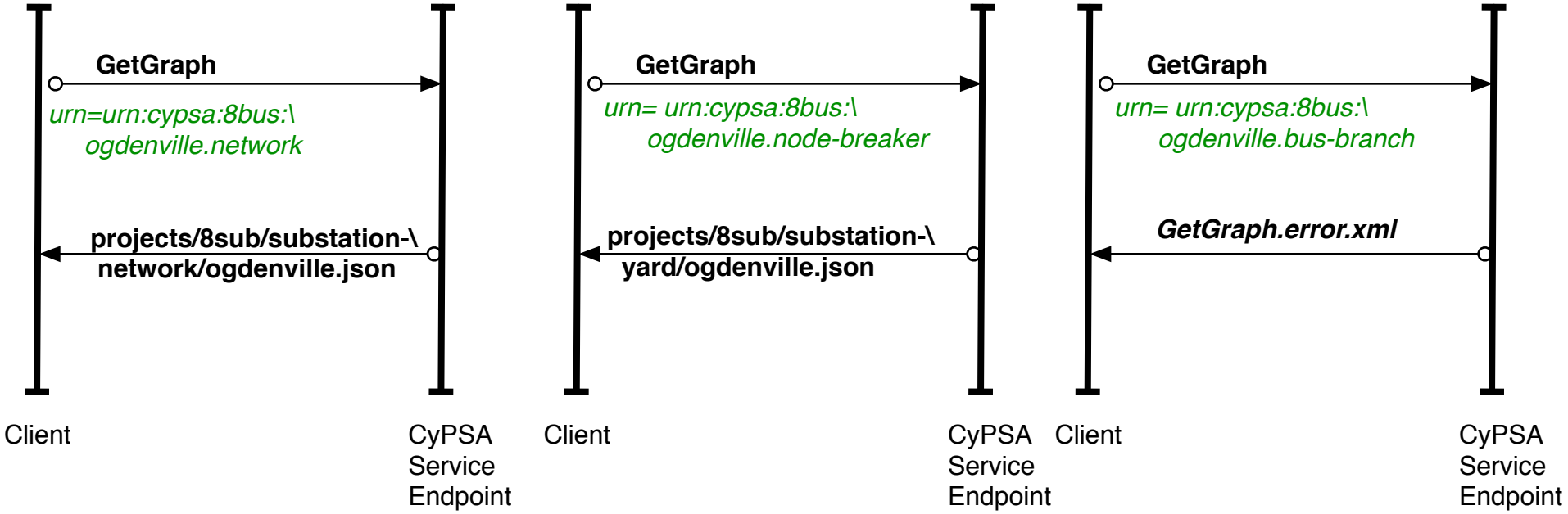


GetGraph

CyPSA Service Catalog

Description	Parameters	Implementation
GetGraph returns a graph representation of the requested work or edition.	*urn:* The CyPSA URN. This should be a work-level URN though other levels could be supported in the future. *format:* JSON Node-Link format by default (suitable for d3.js visualization). RDF Turtle is also possible in the future to support reasoning, however.	This could be accomplished in a variety of ways. I'm curious about a DNS-like service, however. Note that the format may affect the size of the returned request. The request should return a valid graph. Graphs may be validated using the grammars provided on the CPTL-Power ITI GitHub site.

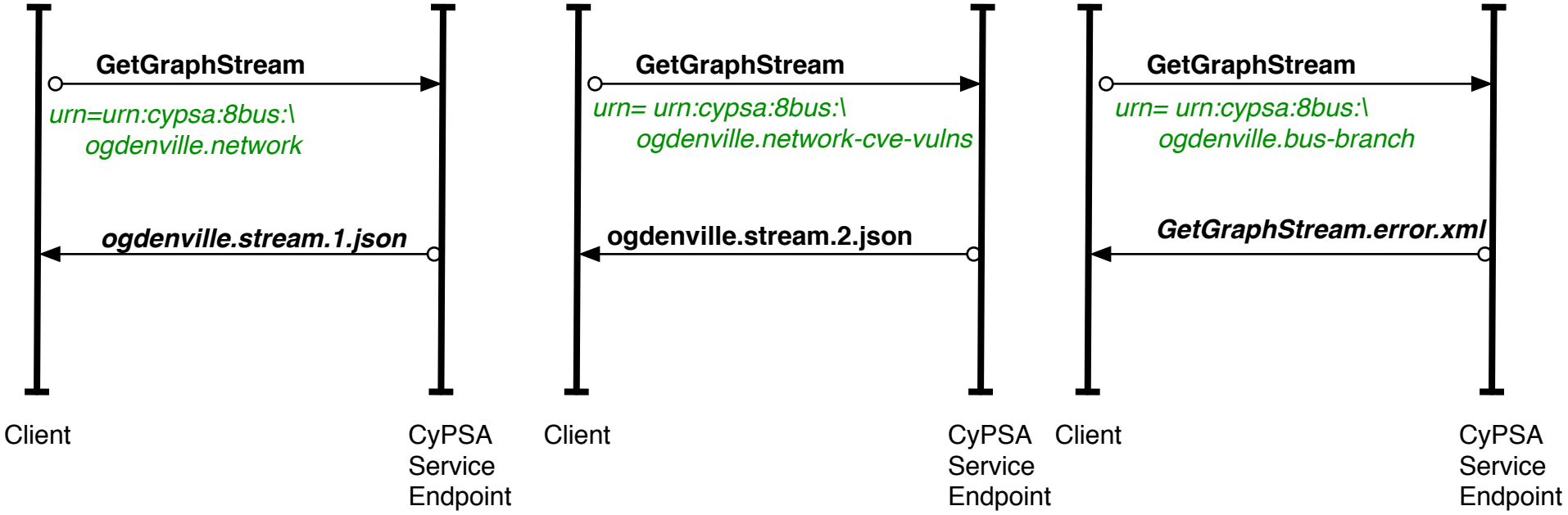
Request and Response



GetGraphStream — *CyPSA Service Catalog*

Description	Parameters	Implementation
GetGraphStream returns a socket across which a streaming graph representation of the requested work or edition will be provided. We will treat the streaming graph as a version/edition of the 'work'. This graph is not stable since it is constantly changing.	*urn:* The CyPSA URN *format:* Returns a simple socket on which to listen. Format to be determined.	This could be accomplished in a variety of ways. I'm curious about a DNS-like service, however. Note that the format may affect the size of the returned request. The request should return a valid graph. We need to choose a good graph streaming API.

Request and Response



Description	Parameters	Implementation
GetGraphJoin returns a graph join of the two graphs requested. The interconnect reference MUST be provided. The definition of graph join is provided in the discussion below.	No parameters are required. There is an optional *config* parameter, however. *urn1,urn2:* The two graphs to join *interconnect:* The interconnect *format:* JSON Node-Link format by default (suitable for d3.js visualization). RDF Turtle is also possible in the future to support reasoning, however.	This could be accomplished in a variety of ways. I'm curious about a DNS-like service, however. Note that the format may affect the size of the returned request. The request should return a valid graph.

Request and Response