XSEDE Information Services "XSEDE-IS" Warehouse Web API Design/Security Description

20 April 2017
Version 0.2

Table of Contents

A. Document History

Relevant Sections	Version	Date	Changes	Author
Entire Document	0.1	2/10/2017	SDIACT-214 Baseline	JP
	0.2	4/20/2017	DSR Revisions	JP

B. Introduction

XSEDE Information Services "XSEDE-IS" enable infrastructure resource discovery and information publishing as described in XSEDE use cases: Subscription (CAN-07), Search (CAN-08), Publish (CAN-11), and Update (CAN-12). To enable discovery a central, reliable, and scalable warehouse aggregates infrastructure information from a variety of information systems like XCDB, RDR, SP managed modules, SP batch schedulers, INCA, NAGIOS, etc. That aggregated information is cross referenced and made available thru a RESTful web API.

The scope of this document is to detail the design and security of the warehouse web API. In this document the term *users* and *developers* are synonymous and refer to both developers and advanced users who need to access information thru a web API.

B.1. Definitions, Abbreviations and Acronyms

- XCDB: XSEDE Central Database
- RDR: Resource Description Repository
- SP: service provider

C. System Overview

XSEDE-IS includes distributed and central components shown in Figure 1 below. Information flows from distributed XSEDE SPs and Information Management Systems such as RDR, Drupal, or Inca to *XSEDE Central Information Services*. This information may flow by push/publish thru the *Pub/Sub Service* or be pulled by *Router Applications* that access remote Information Management Service APIs or data repositories. Central Information Services stores information in an *Information Warehouse* and provides *REST Service(s)* that access the *Information Warehouse*. This document describes the REST Services API. The Pub/Sub service, Router Apps, and the Information Warehouse repository are not in scope for this document[1]. However, to help readers understand the entire system we briefly describe them:

[1] See the "H. References" section

Pub/Sub Service: Enables an information provider to push/publish information into a queue to be forwarded to information consumers/subscribers.

Information Warehouse: Enables persistent storage and access to information. We implemented a data model that supports predefined entities/attributes and dynamic attributes so that many content changes don't require database design changes.

Router Apps: Central programs that either a) access remote information or b) subscribe to information, and then store it in the Information Warehouse.

Information Management Services: XSEDE manages information in RDR, XRAS, Drupal, and other information management systems. These systems are the authoritative source of the information stored in the Information Warehouse. Router Apps access information in these systems thru REST APIs or direct database connections.

The following figure shows some examples of information source, like service providers and information management systems, and information flows from these sources into XSEDE Central Information Services where it can be accessed thru REST Services.

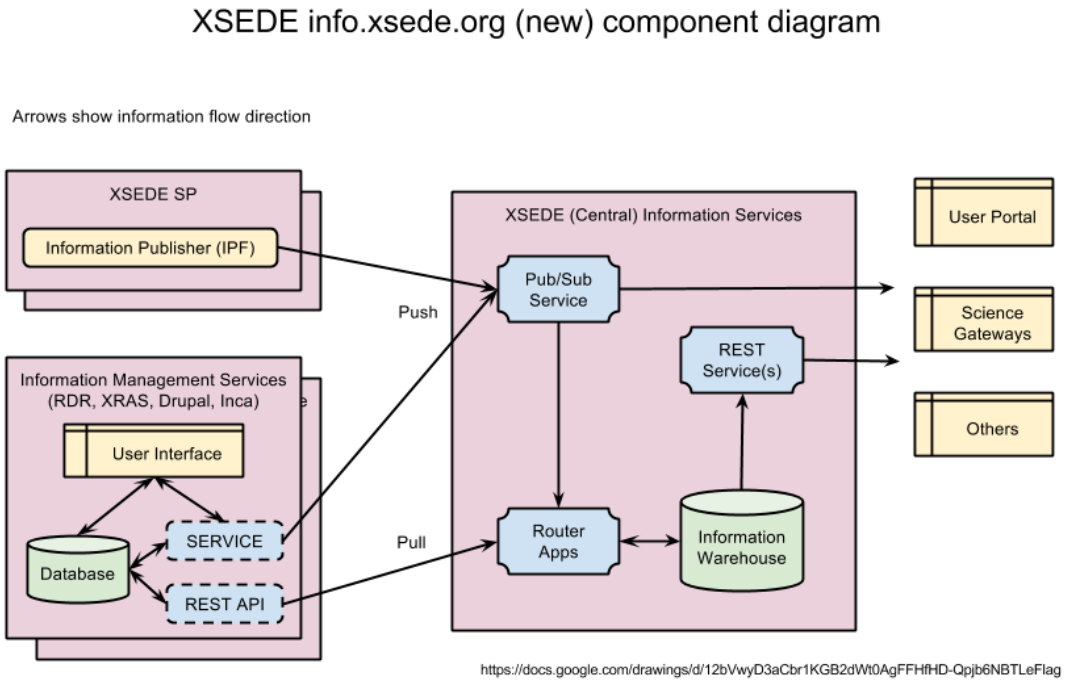


Figure 1: Component and Information Flow Diagram

D. Behavioral Design

From user documentation and from API self documentation users can discover which URLs provide access to different types of Information Warehouse information. Users can use those URLs to access information with a browser, or code them into software for programmatic discovery and information access. All information is available in JSON format, while some of it is also be available in XML, CSV, or browser renderable HTML formats.

E. Design Considerations

E.1. Assumptions and Dependencies

XSEDE users, operators, and staff will generally discover and browse infrastructure information thru the XSEDE User Portal (XUP), the Community Software Repository (CSR), or other browser friendly services. The APIs are not intended for the general user, but for software that needs to programmatically discover and access information and the users/developers that build and support software that access those APIs.

Information in the warehouse originates in a variety of informations systems and needs to be integrated and correlated to make it easily accessible. As a design principle, the users and software accessing the APIs should not have to be concerned with the source of the information or the fact that information came from multiple sources.

E.2. General Constraints

E.2.1. Hardware/Software Environment

The Web API is available thru the info.xsede.org hostname and serviced by redundant info1.dyn.xsede.org and info2.dyn.xsede.org servers. These servers are configured and operated so that the info.xsede.org service will continue to be available when either server is unavailable once the info.dyn.xsede.org dynamic DNS entry is updated. Having physically distributed redundant servers and the ability to quickly switch between them make our target availability possible.

E.2.2. End-user Environment

The end-user (developers and advanced users) environment includes any software or tool that can access RESTful web APIs.

E.2.3. Availability or volatility of resources

The REST Service(s) availability target is 99.5, or less than 44 hours of unavailability a year. Information accessed thru the API needs to be accurate and up-to-date, so it needs to propagate from the source information system to the warehouse as quickly as possible. The volatility/age of specific types of information in the API depends in part on the external sources

of that information and on the components feeding information into the warehouse. The following table details how long it takes information changes to propagate to the warehouse:

Information Type	Latency
SP job status changes	Sub-second/continuous by default (a)
Monitoring test results from INCA/Nagios	Sub-second/continuous from test run (b)
SP batch configuration	1 minute by default (a)
Resource Description Repository (RDR)	5 minutes; will be sub-second in the future
Outages	15 minutes from outage posting
SP software/services	1 hour by default (a)
Project-Resource Map	1 hour
Speedpage	1 hour
XCDB	6 hours

(a) SP configurable
(b) Each INCA/Nagios may have it's own frequency

E.2.4. Standards compliance

The API supports RESTfull HTTP queries compliant with RFC 723x https://www.w3.org/Protocols/.
HTTPS interactions are based on TLS https://tools.ietf.org/html/rfc5246 (replaces SSL).
The API returns data in JSON format, see https://tools.ietf.org/html/rfc7159.

E.2.5. Interoperability requirements

Ingestion interoperability

The warehouse receives information from a variety of information systems. Those information system must either publish the information destined for the warehouse thru the pub/sub service, or provide APIs or interfaces that information services custom "router applications" can use to access the information and store it in the warehouse.

E.2.6. Interface/protocol requirements

This API must support HTTP 1.1.

E.2.7. Data repository and distribution requirements

The API will provide access to the most current available information even when the source system is unavailable. One of the primary functions of the warehouse and REST service(s) is to provide a reliable high-performance front-end to infrastructure information thus insulating users from outages in the originating information system, and insulating the originating information

system from potentially high-frequency user queries. This is accomplished using the Warehouse repository.

E.2.8. Security requirements

All the information in the Query Warehouse is public except for which specific jobs are running on each resource. Read access to non-public information and update (POST/PUT) access are controlled using HTTPS basic authentication with credentials managed by Django. Each unique client that requires authentication will be issued individual username and password credentials.

The TLS certificate for the https://info.xsede.org/ service is:

```
Issuer: C=US, ST=MI, L=Ann Arbor, O=Internet2, OU=InCommon, CN=InCommon RSA Server CA
Subject: C=US/postalCode=61801, ST=IL, L=Urbana/street=1205 W. Clark St, O=University of
Illinois, OU=NCSA, OU=Multi-Domain SSL, CN=info.xsede.org
X509v3 Subject Alternative Name: DNS:info.xsede.org, DNS:info.dyn.xsede.org,
DNS:info1.dyn.xsede.org, DNS:info2.dyn.xsede.org
```

The API's SSL configuration at info.xsede.org receives an A or A+ grade from https://www.ssllabs.com/ssltest/ .

E.2.9. Memory and other capacity limitations

The info1 and info2 servers have ~4 GB of RAM and dual 2 GHz CPUs effectively handling API requests.

E.2.10. Performance requirements

The API needs to be efficient and responsive to queries. It also needs to be reliable and scalable to so that users, software, and services can rely on it to make real-time decisions.

RESTful Services should sustain sub-second GET operation throughput at a rate of 120 operations/minute. Given that POST/PUT operations may do significant processing, we should sustain 5-second POST/PUT operation response at a rate of 12 operations/minute.

E.2.11. Network communications

The API accepts HTTP request over port 80 and HTTPS requests over port 443. HTTPS is recommended for all clients, and required for all clients that have to authenticate.

E.2.12. Verification and validation requirements (testing)

Inca will periodically test/verify Web APIs as follows:

1. The Django Swagger self documentation API hourly
2. The Django admin API daily
3. Each of the APIs in the Component section F.1 below hourly
4. A select subset of APIs for performance daily

The testing plan will verify each of the available APIs.

E.2.13. Usage tracking

All API accesses and errors will be logged and retained for the entire XSEDE 2 program. We anticipate loading log information into an XSEDE logging service when it becomes available.

E.2.14. Other means of addressing quality goals

N/A

F. System Architecture and Detailed Design

The high-level system architecture diagram is shown in Section C above. The Pub/Sub services is based on the RabbitMQ implementation of the AMQP standard. The Information Warehouse is implemented as a PostgreSQL database. Router Apps are implemented primarily as standalone Python programs run periodically via cron or as persistent daemons.
Finally, the REST Service(s) or API are implemented in Django and run under Apache WSGI.

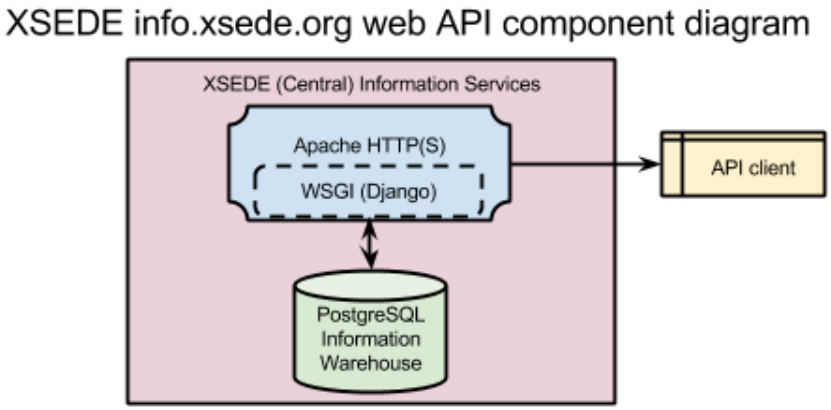


Figure 2: Sub-Component Diagram

The redundant {info1,info2}.dyn.xsede.org servers that host these services are geographically located on IU's gateway hosting environment and at NICS, respectively.

F.1. Component Level Design

The APIs are organized into these groups: General Discovery, Specific Source, and Client Tool categories. The groups and the APIs in each group are documented at:

- https://info.xsede.org/info/restful-api-design
- https://info.xsede.org/wh1/api-docs/

F.1.1. Classification
The API is a web application.

F.1.2. Definition

F.1.3. Responsibilities
XCI is responsible for designing and operating all Information Services. The source information systems that information comes from are maintained by various XSEDE groups.

The API is the primary service enabling programmatic discovery and access of XSEDE federated resource, software, and service information.

F.1.4. Constraints
N/A

F.1.5. Composition
The warehouse and API provide reliable, high-performance, and integrated access to infrastructure information cached and aggregated from a variety of information systems such as XCDB, RDR, SP managed modules, SP batch schedulers, INCA, and NAGIOS.

F.1.6. Uses/Interactions
As shown in *Figure 2: Sub-Component Diagram* the API is used by API clients.

F.1.7. Resources
The RESTful web APIs provide access to XSEDE-IS information resources.

F.1.8. Processing
The APIs provide access to cross-referenced and integrated information about XSEDE federated resources. Information from various sources is processed, transformed, verified, cross-referenced, stored, and then made available thru APIs to clients. Errors that occur during this processing are recorded and made available thru the “processing-status” API. APIs sometimes combine information in different storage objects (tables). Relational database indexes enable efficient cross-referencing and combining of related information into a single API interface.

The information warehouse contains a modest set of resource, software, and service information. Most information types have O(100)s of entities while a few have O(1000)s.

F.1.9. Interface/Exports
Expanding on F.1. above, the following naming conventions are used in the API:

- If the API and associated information mirrors an external repository, the API name is the information type (e.g. outages, rdr-db, speed page, xdcdb)
- If the API is an application specific interface, the API name is the application name (e.g. xdinfo)

- If the API accesses information gathered from a more complex information gathering process, such as subscriptions from multiple publishers, we append suffixes like db-api, provider-api, and views-api to distinguish the role of various API elements.

These APIs return information in a variety of formats. Sometimes it may be close to the same format as the published messages or source repository, while in others it may be in a completely different format because it combines information from multiple sources.

F.2. Data Design

XSEDE-IS is designed to automatically ingest, warehouse, and provide API access to an evolving set of information. As the source information systems change, the warehouse will automatically ingest information about new infrastructure resource, and new attributes about that infrastructure. As information is ingested subset of fields are required and expected in order to properly index and cross-reference the information. Additional field that may appear over time will automatically be accepted, stored in the warehouse, and made available thru the API. The warehouse achieves this by storing all fields that it doesn't recognize in a JSON blob in the database.

The primary identifiers used to identify which resource information is about, and also used to cross reference information in the warehouse are:

- ResourceID (a.k.a info_resourceid in RDR): A human readable unique resource ID
- SiteID: the site or institution operating a resource
- AppName: the human readable unique name for a piece of software
- InterfaceName: the human readable unique name for a type of network service

Examples:

ResourceID = bridges.psc.xsede.org
SiteID = psc.xsede.org
AppName = abaqus
InterfaceName = org.globus.gridftp

F.3. Delivered Packages

Planned Package Formats	**Target Audience**
REST Service(s) at https://info.xsede.org/	Anyone needing to discover and access infrastructure information

G. Interface Design

Summary of external interfaces that the API relies on.

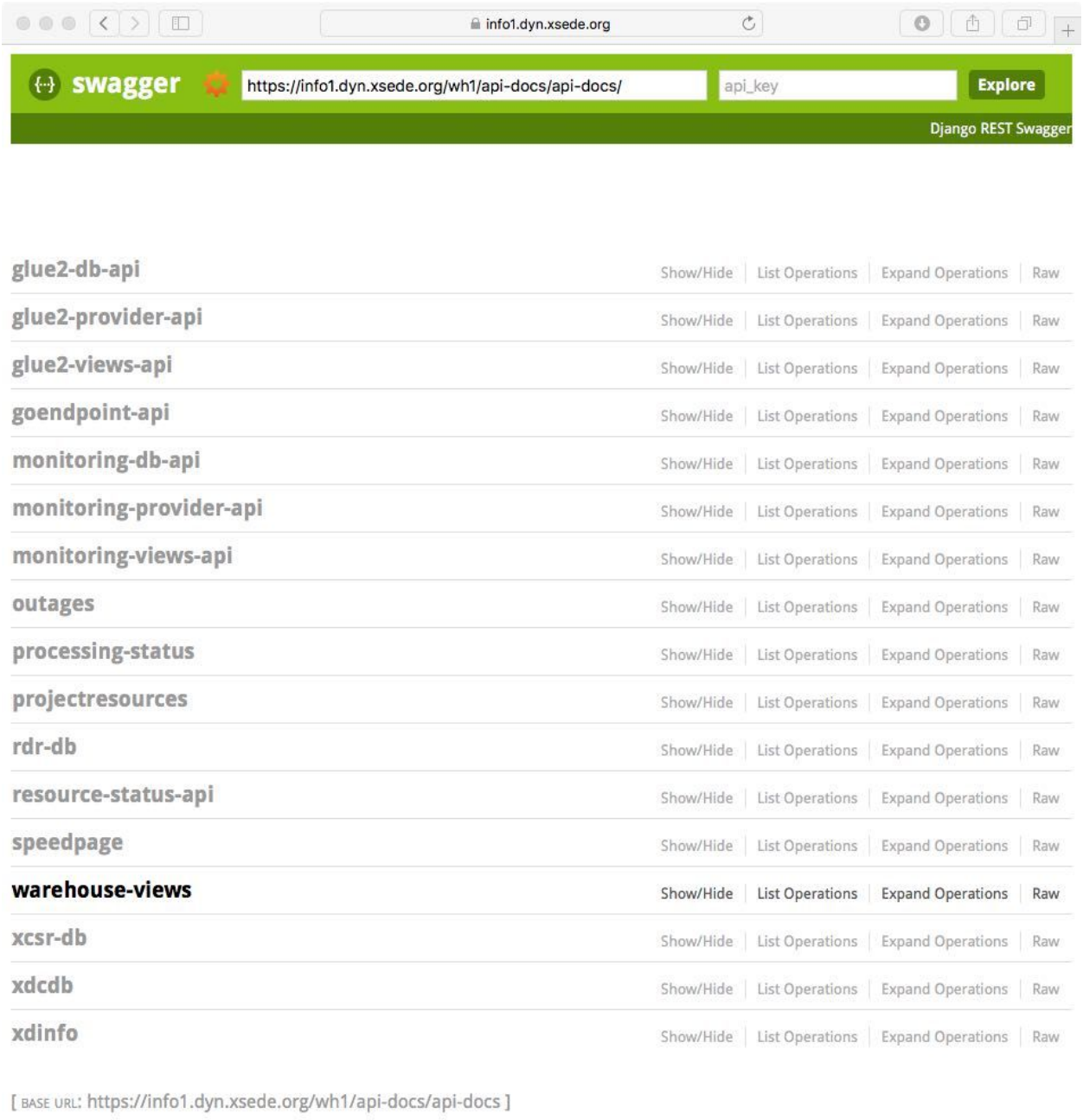
External Interfaces	How used
RDR	The API accesses information obtained from RDR.
XCDB	The API accesses information obtained from XCDB.
IPF	The API accesses information published by SPs using IPF.
Speedpage	The API accesses information obtained from the Speedpage database.
Inca/Nagios	The API accesses information obtained from Inca and Nagios.

The interfaces provided by this API are referenced in F.1. above.

G.2. Use Documentation

See F.1. above.

The following screenshot shows the main API self-documentation page generated with Swagger. From this page developers can explore the API endpoints, the REST operations each API support, and any parameters that are part of the API.



G.3. Training

The web API is self documenting. No additional training is offered.

H. References

[1] XSEDE API Documentation: https://portal.xsede.org/xsede-api