

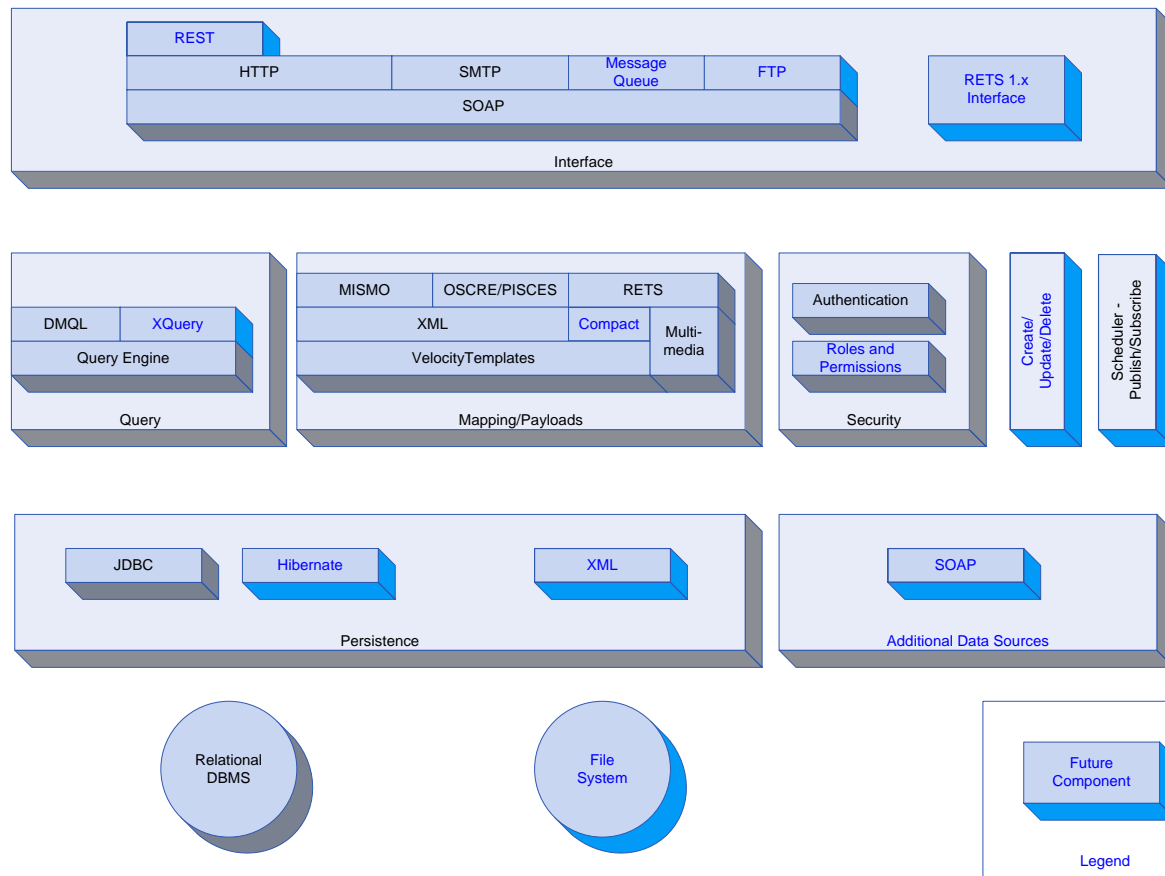
**RETS 2 Engineering
August 2006 Chicago**

RETS2 Design - RETS as a Meta-Standard

- ◆ A standard that deals with other standards
 - Discovery
 - Metadata
 - Transport
 - Mappings

RETS2 Framework - Server

RETS2 Server Architecture
Service Bus



Payload/Output Mapping - Velocity

- ◆ Velocity is a Java-based template engine. It permits anyone to use a simple yet powerful template language to reference objects defined in Java code.
- ◆ Can be used to generate XML from templates
- ◆ Common uses of Velocity: placeholders for dynamic information, source code generation, XML transformation
- ◆ References in XML work with a context object from the server
- ◆ Provides basic control and conditional statements
- ◆ Call Java methods, include files, and create macros

Payload/Output Mapping – Velocity - nVelocity

- ◆ Ported from Apache Velocity Framework
- ◆ Uses the same Velocity Template Language
- ◆ Able to reuse Java Velocity templates with no change
- ◆ Used by C# and VB.NET
- ◆ Uses nAnt and log4net

Manifest

- ◆ Opaque data model
- ◆ Flexible - determine mimeType from contentType
- ◆ Content can carry RETS schemas, images, ANY TYPE of "payload"
- ◆ Use manifest type as SearchResponse and UpdateRequest

Interface – SOAP 1.2 – XFire

- ◆ Codehaus XFire is an implementation of the SOAP 1.2 framework for constructing SOAP processors such as clients, servers, gateways, etc.
- ◆ Benefits of the XFire implementation:
 - **Speed** – Streaming XML for high performance (2.5 X faster than Axis)
 - **Flexibility** – Pluggable bindings for JAXB and XMLBeans
 - **Proven interop with MTOM and Microsoft's WSE3**
 - **Spring Framework**

Business API Facade

- ◆ Façade: defines a higher-level interface that makes a sub-system easier to use
- ◆ Hides RETS-specific implementation details: DMQL, metadata
- ◆ Over-loaded methods – handle data in multiple formats (stream, etc.)
- ◆ Providers persistence:
 - Data Persistence
 - Metadata Persistence
- ◆ Formats request, parses response; Data and Metadata Management
- ◆ Lowers barrier of entry for RETS client developers
- ◆ Encapsulates transport layer – handles SOAP (RETS2) and HTTP (RETS 1.x)

MTOM – Message Transport Optimization

- ◆ MTOM is the recommended method for sending attachments in SOAP 1.2 replacing SwA (SOAP with Attachments) and DIME.
- ◆ The optimization in MTOM is that the attached file is not encoded at all. It is sent as the raw bytes across the network.
- ◆ If the content needs to be base64 encoded to allow for signature or gross message level encryption (not element level), the encoding only happens at each of the endpoints were the 4/3 data expansion is less of a factor.
- ◆ WSE 3.0 is still XmlDocument class based internally and so all messages will be loaded into memory so you can never achieve complete streaming end to end. (Other than when written to the network stream for MTOM).
- ◆ MTOM enables a file to be directly written to the network stream by implementing IXmlSerializable

Integrating RQL

- ◆ Using ANTLR and CRT's rql.g
- ◆ ANTLR can generate Java and C# versions of the Parser and Lexer Code
- ◆ The ANTLR parser returns a parse tree which is walked to generate the WHERE clause that is passed to the Data layer
- ◆ Per implementation - ties into the metadata

SOAP Faults – Handling Errors in RETS2

- ◆ Code – Generally will be the Sender or the Receiver Codes
- ◆ Subcode - RETS specific faults
- ◆ Reason - RETS fault messages
- ◆ Custom subcodes

SOAP Faults - Example

```
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope"
  xmlns:retserr="http://www.rets.org/faults"
  xmlns:xml="http://www.w3.org/XML/1998/namespace">
  <env:Body>
    <env:Fault>
      <env:Code>
        <env:Value>env:Sender</env:Value>
        <env:Subcode>
          <env:Value>retserr:InvalidResource</env:Value>
        </env:Subcode>
      </env:Code>
      <env:Reason>
        <env:Text xml:lang="en">
          The Resource "RES" is not supported by this Provider.
        </env:Text>
      </env:Reason>
    </env:Fault>
  </env:Body> </env:Envelope>
```

Extensibility

- ◆ Extensibility points in Metadata
- ◆ Extensibility points in WellKnown Payloads
- ◆ Add extensibility points to custom schema

Metadata – Resource List

- ◆ Metadata Formats - information about the resource's data
 - ◆ OutputFormats - how search results are returned
 - ◆ InputFormats - how updates are accepted
 - ◆ Vocabularies - how to query the resource
 - ◆ Display transformations

Metadata - Vocabulary

- ◆ Fields to construct a query
- ◆ Required Fields
- ◆ DisplayGroups
- ◆ WellKnown, Local – and functional vocabularies
- ◆ Query language separates query from OutputFormat