Raytheon

Customer Success Is Our Mission

AWIPS Continuous Technology Refresh (CTR)

AWIPS Software CTR

T010 Silver Spring TIM

Dec. 4, 2008

This document includes data that shall not be duplicated, used, or disclosed – in whole or in part – outside the Government for any purpose other than to the extent provided in contract DG133W-05-CQ-1067. However, the Government shall have the right to duplicate, use, or disclose the data to the extent provided in the contract. This restriction does not limit the Government's right to use information contained in this data if it is obtained from another source without restriction. The data subject to this restriction are contained in all sheets.

AWIPS-II T010 TIM Topics

- ESB transition from MULE to apache CAMEL
- T010 Architecture changes that impact developers
  - Serialization Enhancements
  - EDEX Plug In Enhancements
  - Metadata schema and purging Enhancements
  - Data cube and derived parameter enhancements
  - Subscription Replacing textdb trigger capability
- SmartTools in T010
- NCLADT Q&A discussion
- General Alerting in T010 (Guardian)
- NAWIPS transition
- Hydro Issues
  - GUIs and interfaces
  - RiverPro
- Q&A

NWS TIM Questions – Additional Clarification
ESB upgrade

Raytheon

CAMEL

- Mule 1.4.X community tree appears to be dying and has a large number of unresolved problem reports against it
  - Major problem for AWIPS-II is a bug in the JMS reconnection implementation that effects AWIPS-II stability
- Mule 2.X Mule Enterprise has a more restrictive license and has a new XML schema for wiring components
- During T010 we evaluated various versions of Mule and other ESBs
- At the time of the TIM we had decided that “Apache Camel 1.5” had the most potential for T010 after getting unsatisfactory results from Mule

NWS TIM Questions – Additional Clarification
ESB upgrade Status Nov 18, 2008

Raytheon

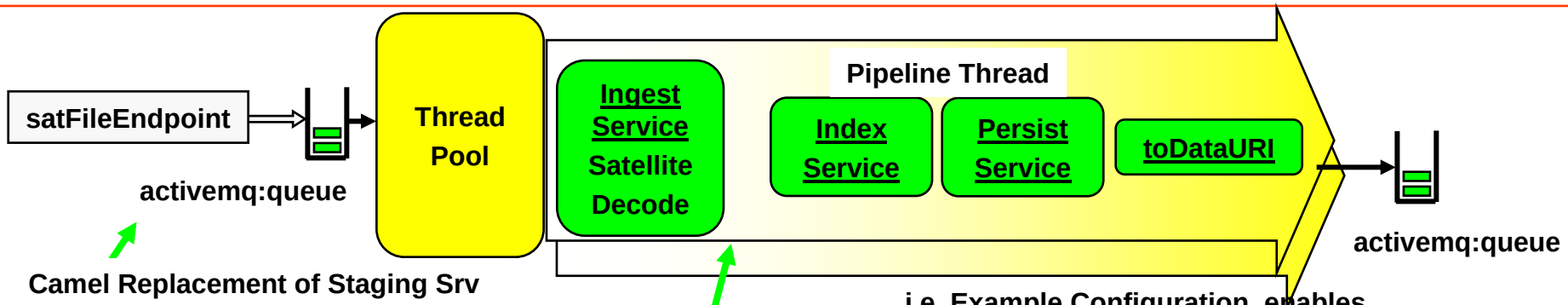
CAMEL

- Apache Camel has been evaluated under heavy data loads from all the plug-ins except for 4
  - Handles large spikes in data flow without memory overruns
  - Better messaging efficiency
    - activeMQ and the ESB run in the same JVM
    - JMS automatically switches to virtual memory queues for internal messages
    - Greater control of ingest processing through execution pipelines
    - Better management of threads through thread pools with less thread context switching
    - Simplified startup (only one JVM process instead of two)
  - Uses the SPRING dependency injection container
  - AWIPS-II services become simple POJOs without any ESB dependencies
  - Simplified XML ESB wiring
    - Wiring XML packaged within the plugin JAR file
  - http://activemq.apache.org/camel/index.html
- Log4j will continue to be used but the location of log files will change

Camel ESB Wiring Packaged in Plug In

Simplies plug in deployment

CAMEL



Camel Replacement of Staging Srv

i.e. Example Configuration, enables adaptive configuration

Example: Satellite Ingest

```
<!-- Begin Sat routes -->
<route id="satFileConsumerRoute">
    <from ref="satFileEndpoint"/>
    <bean ref="fileToString" method="absPath"/>
    <setHeader headerName="pluginName">
        <constant>satellite</constant>
    </setHeader>
    <to uri="activemq:queue:Ingest.Generic"/>
</route>
<route id="satIngestRoute">
    <from uri="directvm:satelliteIngest"/>
    <multicast parallelProcessing="false">
       <try>
          <pipeline>
             <bean ref="stringToFile" method="toFile"/>
             <bean ref="satDecoder" method="decode"/>
             <to uri="directvm:persistIndexAlert"/>
          </pipeline>
          <catch>
             <exception>java.lang.Throwable</exception>
             <to uri="log:sat?level=ERROR&amp;showBody=fal
          </catch>
       </try>
    <bean ref="deleteFile" method="delete"/>
    </multicast>
</route>
```

- Pipeline out of thread pool decreases thread switching increasing performance
- Thread Pool enable fine grain tuning and enables high priority data to pass through

CAMEL ESB Monitoring

Built in Web based and JMX monitoring

CAMEL

http://awips-dev2:8080/activemqAdmin/queues.jsp

ActiveMQ

Home | Queues | Topics | Subscribers | Send

Queue Name Create

Queues

Name	Number Of Pending Messages	Number Of Consumers	Messages Sent	Messages Received	Views	Operations
Ingest.Bufr	0	2	16524	16524	Browse atom rss	Send To Purge Delete
stageNotification	193	0	0	0	Browse atom rss	Send To Purge Delete
Ingest.Generic	2419	6	313950	311531	Browse atom rss	Send To Purge Delete
edex.ProductSrvRsp	0	0	0	0	Browse atom rss	Send To Purge Delete

ActiveMQ

The Apache Software Foundation http://www.apache.org/

Home | Queues | Topics | Subscribers | Send | Support

Browse Ingest.Generic

Message ID	Correlation ID	Persistence	Priority	Redelivered	Reply To	Timestamp	Type	Operations
ID:awips-dev2.oma.us.ray.com-41417-1228091569957-2:3182:1:1:8493		Persistent	4	false		2008-12-01 15:52:45:063 GMT		Delete
ID:awips-dev2.oma.us.ray.com-41417-1228091569957-2:3182:1:1:8499		Persistent	4	false		2008-12-01 15:52:45:545 GMT		Delete

- Graph
- XML

Useful Links

- Documentation
- FAQ
- Downloads
- Forums

TO10 Serialization Improvements

- Why?
  - JiBX doesn't work particularly well in our plugin architecture. To build stand-alone plugins, copies of all dependencies must be kept around with the pre and post JiBX classes.
    - No more post compilation steps
  - Performance: possible to do even better than JiBX
  - Numerical Payloads: extremely inefficient in XML, much more efficient in Binary payloads

TO10 Serialization Improvements

- Proposal:
  - Replace the existing JiBX serialization technology with a two pronged approach
    - XML Serialization using JAXB 2.1
    - Binary Serialization using Thrift
  - Implement both technologies purely at runtime using state-of-the-art technologies to maintain fast performance
    - No post-compile steps
    - Descriptions of serialization implemented purely in Annotations
  - Use binary in communication between CAVE and EDEX, XML in user-presented situations

TO10 Serialization Improvements

- Why JAXB 2.1?
  - Included in JDK 6
  - Has EXTENSIVE built in class support for HashMaps, Lists, etc. (better support than JiBX)
  - Like JiBX, works with your POJOs, not class generation
  - Other considered projects: XStream, XMLBeans, others.

TO10 Serialization Improvements

- Why Thrift?
  - Thrift is open standard. Developed by Facebook, currently an Apache Incubator project.
  - Simple and easy to understand.
  - API available in variety of languages
  - Was easily modified to become “self-describing”
  - Other considered projects: Google Protocol Buffers (had compilation requirements), NetCDF4/HDF5 (not suited for wire protocols), Fast Infoset (Performance concerns)

TO10 Serialization Improvements

- Move to annotations simplifies coding experience. Example:

```
@XmlAccessorType(XmlAccessType.NONE)
@DynamicSerialize(type="Thrift")
public class FooRecord {

    @XmlAttribute
    @DynamicSerializeElement
    private Integer resolution;

    @XmlElement
    @DynamicSerializeElement
    private String icao;
...
}
```

TO10 Serialization Improvements

- Limitations:
  - Must always have default constructors
  - Must have setters/getters unless you want to write your own factory class

TO10 Serialization Improvements

- Thrift Serialization Modfications
  - Make Thrift “self-describing” which makes it possible to reconstruct java objects without having an exact template of what the object will look like. Out of the box, projects like Thrift (and Protocol Buffers) require a compilation step.
    - Makes things more dynamic
    - No post-compile requirements
    - No writing of adapters between generated objects and POJOs

TO10 Serialization Improvements

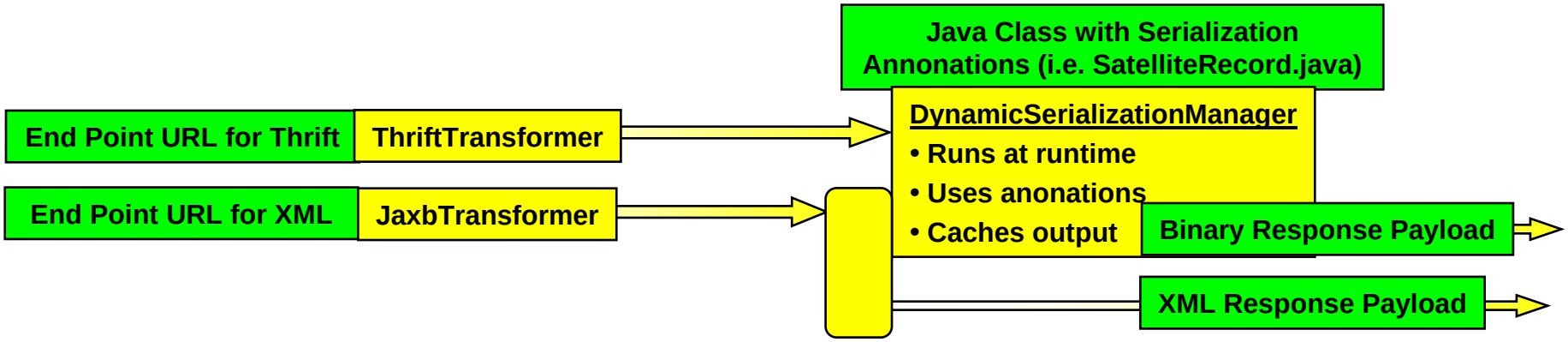
- To facilitate faster marshalling and unmarshalling, the reflection in the custom Thrift marshaller is enhanced using CGLib
  - CGLib is used by Hibernate
  - Does on the fly code generation to make Reflection operations fast
- Discovery facility discovers serializable objects and caches for performance

TO10 Serialization Improvements

- Preliminary performance benchmarks
  - Test #1: Serialize and Deserialize 1000 “typical” MetarRecords
    - JiBX: 752ms Serialize / 1090ms Deserialize
    - Dynamic Thrift: 550ms Serialize / 801ms Deserialize
  - Test #2: Serialize and Deserialize Object with float[] of size of grid218.
    - JiBX: 240ms Serialize / 310ms Deserialize
    - Dynamic Thrift: 42ms Serialize / 38ms Deserialize

Service Message Serialization

SOA Dynamic Binary Serialization Increases Performance




```
<camelContext id="uengine-camel" xmlns="http://activemq.apache.org/camel/schema,
    <!-- UEng
    <route id
        <from uri="jettyedex:http://0.0.0.0:9581/services/pyproductthrift" />
        <bean ref="uEngine" method="executePython" />
        <bean ref="serializationUtil" method="transformToThrift" />
    </route>
            <route id="uEngineHttpJaxb">
            <from uri="jettyedex:http://0.0.0.0:9581/services/pyproductjaxb" />
        <bean ref="uEngine" method="executePython" />
        <bean ref="serializationUtil" method="marshalToXml" />
            </route>
            <route id="uEngineJmsThrift">
        <from uri="activemq:queue:uEngineThrift" />
        <bean ref="uEngine" method="executePython" />
        <bean ref="serializationUtil" method="transformToThrift" />
    </route>
            <route id="uEngineJmsJaxb">
        <from uri="activemq:queue:uEngineJaxb" />
        <bean ref="uEngine" method="executePython" />
        <bean ref="serializationUtil" method="marshalToXml" />
    </route>
```

<<CAMEL ESB Wiring>>

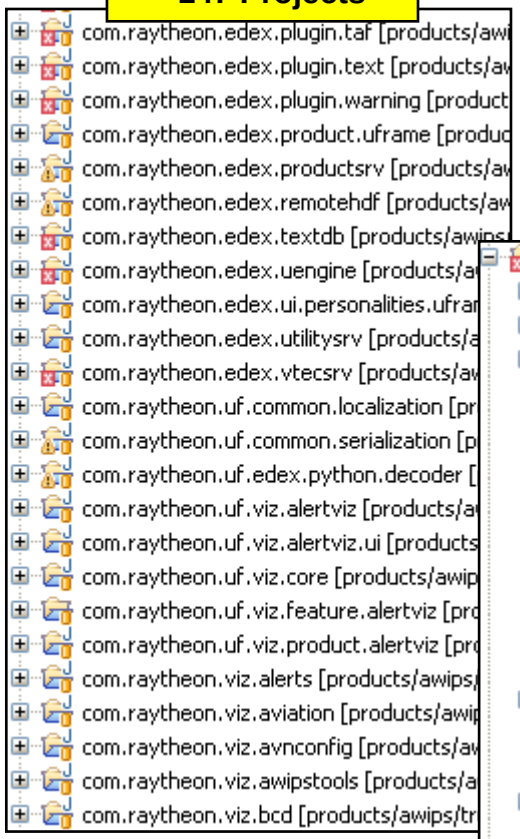
Exposed to Developer through ESB transformer Pattern

- Thrift is normally static and runs at compile to produce java classes
- T010 added dynamic serialization based on annotations for greater control of data that is transferred
- T010 added caching of generated objects so that the generation of classes at a one time hit

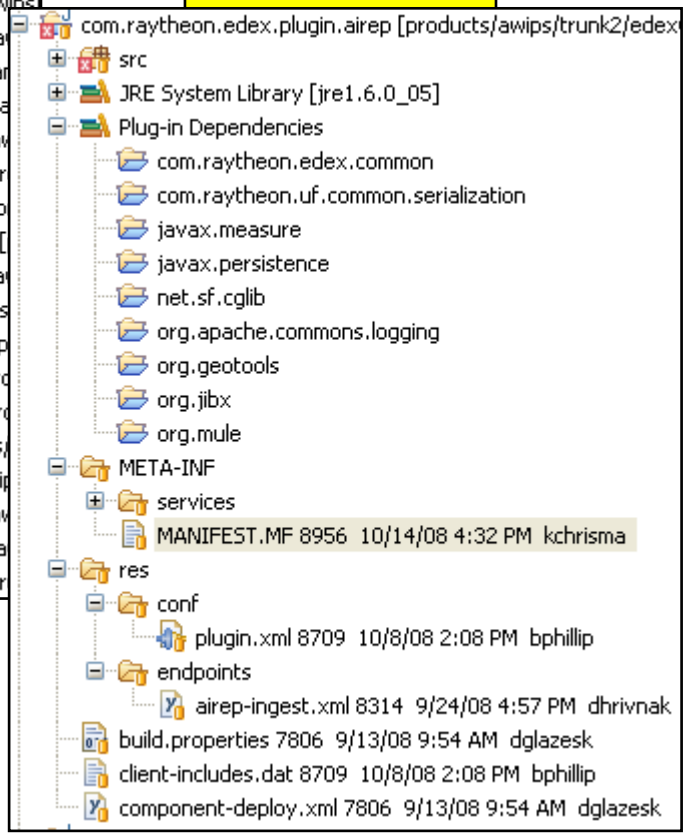
SOA datatype Plug In Enhancements

Improved Standardization using OSGi model

<<OSGi plug in>>
~147 Projects



<<OSGi plug in>>
OSGi structure



- **Enables auto build of EDEX projects when with CAVE changes**
- **Allows Eclipse tools to work across EDEX and CAVE**

SOA datatype Plug In Enhancements

Field extendable decoder model using Python

i.e. Edex Warning Plug In

- Enables reuse of GFE "VTECdecoder.py"
- Enables decoder changes and extensions to deployed systems

EDEX Plug In Core Interfaces

Java Interface To PythonDecoder

PythonNumpyByteArray

WarningDecoder.py

WarningRecord

```
@Entity
@Table(name="warning")
@XmlRootElement
@XmlAccessorType(XmlAccessType.NONE)
@DynamicSerialize
public class WarningRecord extends PluginDataObject {

    private static final long serialVersionUID = 1L;

    @Column(length=32)
    @XmlAttribute
    @DynamicSerializeElement
    private String wmoid;
```

<Metadata Table> Warning

Note: Serialization Annonations

SOA datatype Plug In Enhancements

Simplified metadata definitions through Java annonations

Raytheon

Arch - EDEX Plug In

Java Plugin Source Code + annotations → **MetaData Table Creation**

i.e. Example

SatelliteRecord.java

```
@Entity
@Table(name = "satellite")
@XmlRootElement
@XmlAccessorType(XmlAccessType.NONE)
@DynamicSerialize
public class SatelliteRecord extends PluginDataObject
        ISpatialEnabled {

    private static final long serialVersionUID = 1L;

    /**
     * The source of the data - NESDIS
     */
    @Column(length=31)
    @DataURI(position = 1)
    @XmlAttribute
    @DynamicSerializeElement
    private String source;
```

satellite Table in Metadata DB

datauri [PK] character varying(255)	in	bt	cli	cr	cli	cli	cli	cr	in	cr	in	re	rea	cr	siz int	source character var	ur ch	up re	up re	cc in
/satellite/2008-11-25_09:22:15.0/NESDIS/Mi	0	FA	20	[]	20	20	20	Mi	43	Re	0	0	0	Al	504	NESDIS		0	0	-16

Hibernate Annotations
Note: Same style as JavaDoc

- **Using Hibernate annotations eliminates two XML files from each plugin**
- **Simplifies the creation / maintance of plugin metadata**

Simplified metadata storage

More flexible metadata purging

Arch - EDEX Plug In

name [PK] characte	database character var	initialized boolean	retentiontime integer	tablename character var	version real
airep	metadata	TRUE	24	airep	1
binlightning	metadata	TRUE	24	binlightning	1
bufrmos	metadata	TRUE	24	bufrmos	1
bufrua	metadata	TRUE	24	bufrua	1
ccfp	metadata	TRUE	24	ccfp	1
gfe	metadata	TRUE	24	gfe	1
goessounding	metadata	TRUE	24	goessounding	1
grib	metadata	TRUE	24	grib	1
modelsounding	metadata	TRUE	24	modelsounding	1
obs	metadata	TRUE	24	obs	1
pirep	metadata	TRUE	24	pirep	1
poessounding	metadata	TRUE	24	poessounding	1
profiler	metadata	TRUE	24	profiler	1
radar	metadata	TRUE	24	radar	1
recco	metadata	TRUE	24	recco	1
redbook	metadata	TRUE	24	redbook	1
satellite	metadata	TRUE	24	satellite	1
sfcobs	metadata	TRUE	24	sfcobs	1
shef	ihfs	TRUE	24	<unspecified>	1
taf	metadata	TRUE	24	taf	1
text	fxa	TRUE	24	<unspecified>	1
warning	metadata	TRUE	24	warning	1

Quart Timer Once / hour

PurgeSrv

<java>
DefaultPurgerImpl

<java>
GfePurger

GFE Plug In

<plugin registry>
Plug_Info Table

ESB implementation of Metar2Shef

Leverages ESBs transformer and endpoint routing

DataCubeContainer Enhancements

DataCubeContainer, Plug In Enhancements, 3D/Slice Method

Existing Implementation:

The existing DataCubeContainer is GRIB specific:

- Container is GRIB specific
- GRIB metadata is also managed by the container
- Point and cross section based parameters are not supported
- Auto update of derived parameters is not supported

The existing DataCubeContainer maintains the GRIB metadata for performance reasons:

- Instead of performing a catalog query each time the volume browser is opened, the grid metadata is read in from memory
- The metadata is also used to determine which derived parameters are available and stores those in memory as well
- This special case will be maintained, although the logic will be moved into the plugin

DataCubeContainer Enhancements

New Implementation:

Here is what is proposed

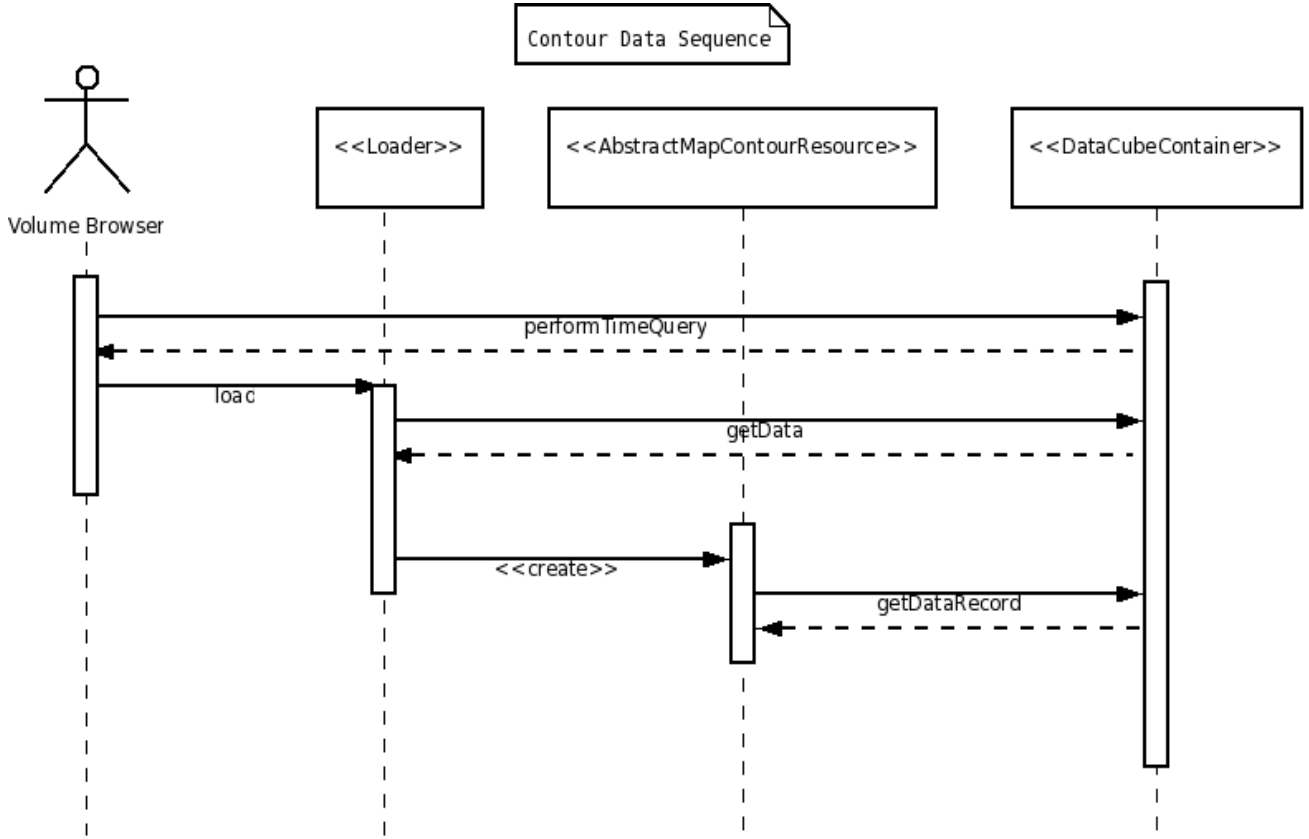
- Point and line based derived parameters needs to be implemented within the container
- GRIB specific code will be moved into the GRIB plugin
- Missing capabilities including the 3D cube, the Slice method, and level definition support will be implemented to complete derived parameter capability
- Additionally, the static calculations for dx, dy, and Coriolis need to be completed and verified

DataCubeContainer Enhancements

DataCubeContainer Data Sequence:

The existing DataCubeContainer data sequence will not change:

- The data chooser will perform a time query
- The collected metadata will be passed to the loader which will then create an instance of the requested resource
- The resource will retrieve the required data from the DataCubeContainer



Datatype Plugin Enhancements

New Implementation:

Here is what is proposed

- At startup, the IPluginDataAccess implementations packaged with each data type plugin will be found
- The discovery method will follow the pattern used by other packages in com.raytheon.viz.core such as Units
- An interface will be created that a plugin can implement which, when present, will allow the container to created derived products for that plugin data type
- Additionally, the interface will include a metadata translator that will format the metadata in such a way as to be usable by a data chooser widget

Datatype Plugin Enhancements

IPluginDataAccess Interface

- performTimeQuery – returns all DataTime objects valid for the query parameters
- getData – return all PluginDataObjects valid for the layer property passed
- getPoints – an overloaded method that takes either a list of stations or a list of points and returns values for those locations
- getLine – another overloaded method that takes start/end points or start/end stations and returns value for those locations
- getMetadata – a method that will return the stored metadata in a way meant to be displayed by data chooser interfaces (Volume Browser)

```
<<interface>>
IPluginDataAccess
+performTimeQuery(queryParams:Map<String,
                  RequestConstraint>): DataTime[]
+getData(property:LayerProperty): List<Object>
+getPoints(stations:List<String>,
           property:LayerProperty): List<Object>
+getPoints(points:List<Point>,
           property:LayerProperty,
           levels:List<Level>): List<Object>
+getLine(startStation:String,endStation:String,
         property:LayerProperty,
         levels:List<Level>): List<Object>
+getLine(startPoint:Point,endPoint:Point,
         LayerProperty:property,
         levels:List<Level>): List<Object>
+getMetadata(): GridTree
```

Datatype Plugin Enhancements

New Implementation For GRIB:

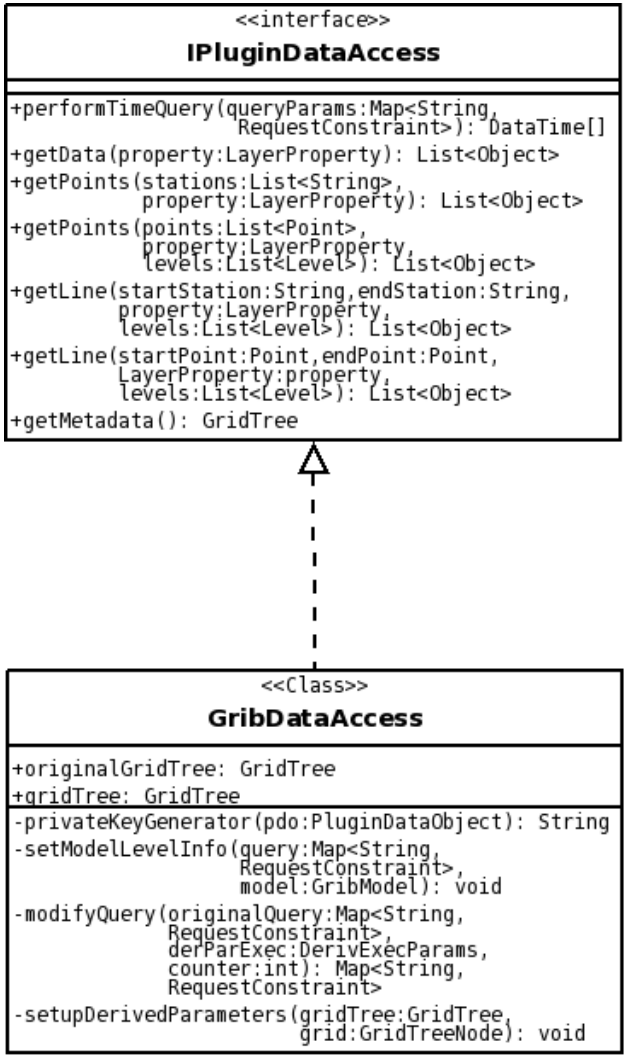
Here is what is proposed

- All GRIB specific code will be removed from the DataCubeContainer and moved into the GribDataAccess class
- GridTree will be modified (and renamed) to return a new object similar to GridTree but more universal
- GRIB query performance will be revisited to see if it is required to keep the entire grid tree in memory as it exists now

Datatype Plugin Enhancements

GribDataAccess Class

- The GribDataAccess class will implement the functionality that was previously run from within the DataCubeContainer. This includes support methods to properly populate the clean grid tree with derived parameters.



3D/Slice Method and Loose Ends

The Question Of The 3D Cube And Slice Method:

3D Cube and Slice Method:

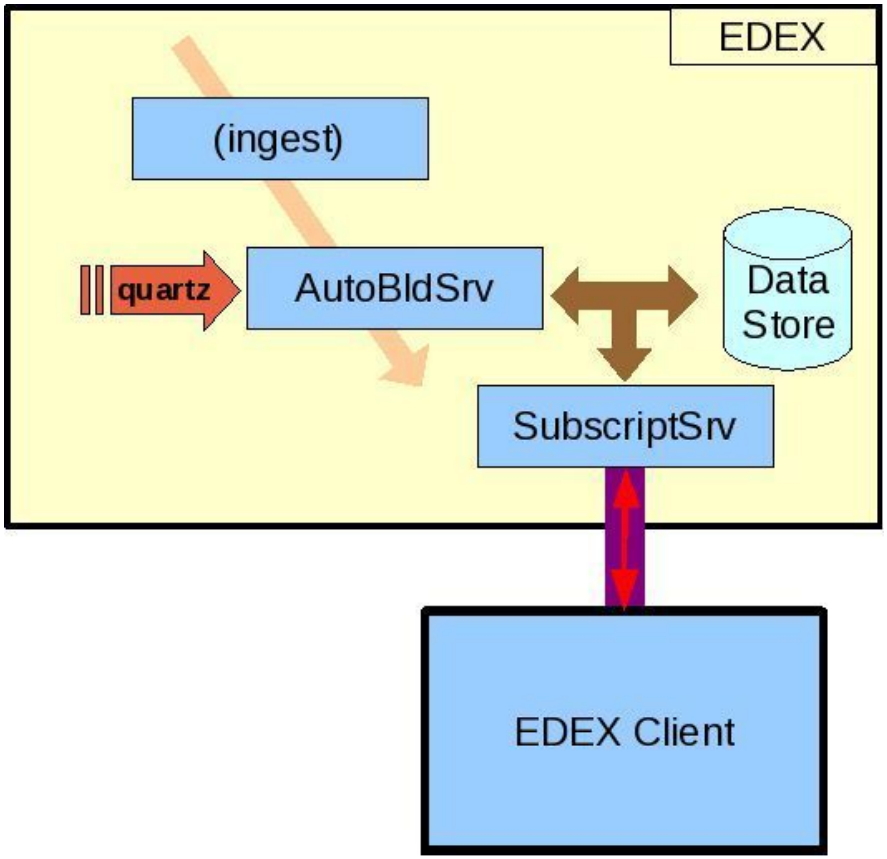
- They are a similar capability a line request which is already present in the data cube within CAVE
- It provides the ability to retrieve levels that are not normally available from a model run through interpolation
- This capability presently requires direct access to the HDF5

The Move To Grid Data Returning In Response:

- The current path for data requests includes moving away from NFS mounts and access to HDF5
- When this happens, the data cube will need to be moved to the server for point and line requests
- If this occurs, then the new 3D/Slice capability should be moved there as well and turned into a uEngine capability
- In that case, new development would be needed to satisfy this functionality

Subscription Service Design Concept

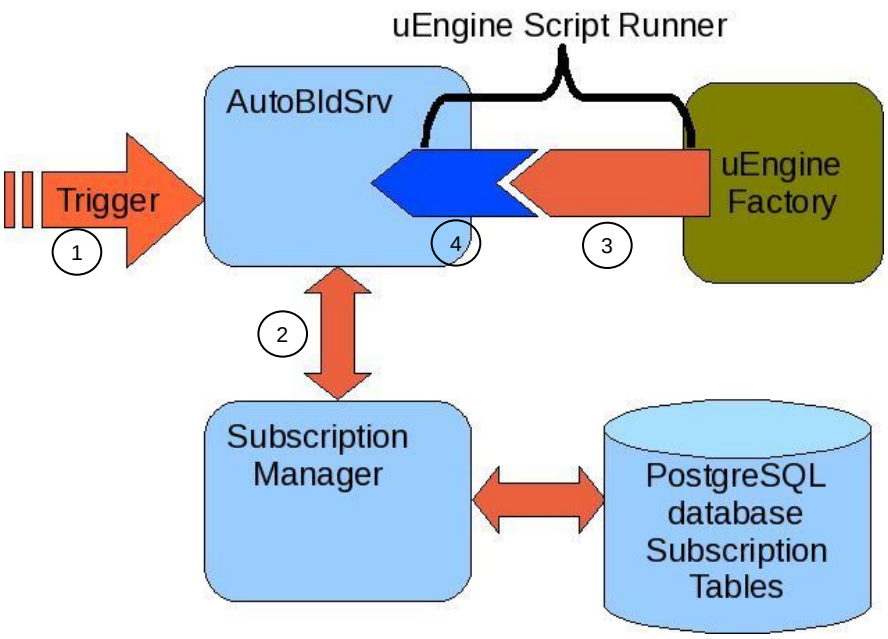
- This diagram shows the revised design of the subscription management functionality
  - the Subscription Manager is merged into NotifySrv and renamed SubscriptSrv
  - an EDEX client interacts directly with SubscriptSrv to manage subscriptions
  - SubscriptSrv interacts with the data store and passes messages to the script runner (AutoBldSrv)
  - AutoBldSrv will be refactored to allow multiple script runners as well as product and timer based execution



Subscription Service Design Concept

Subscription Fulfillment

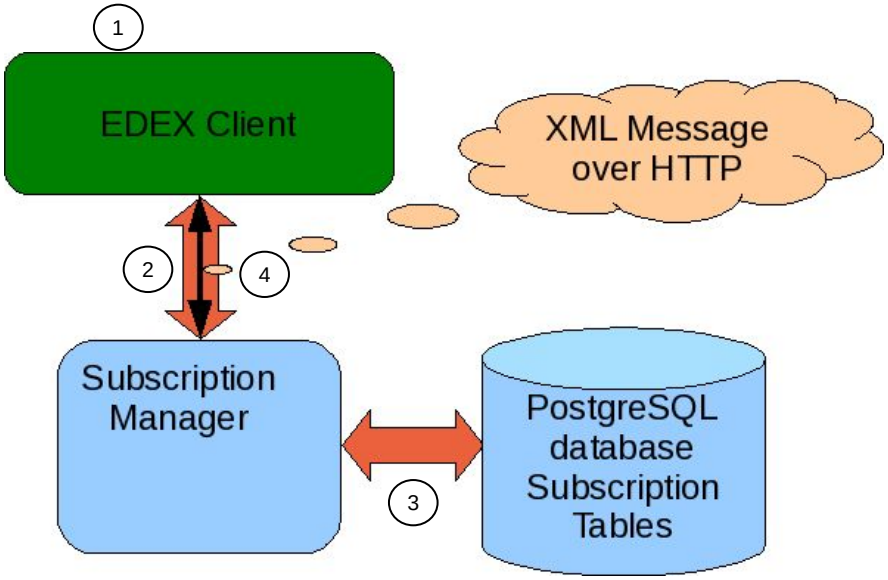
1. AutoBldSrv is triggered by system
2. AutoBldSrv obtains subscription list from Subscription Manager
   - Subscription Manager obtains subscription list from database
3. AutoBldSrv obtains script runner from uEngine factory
4. AutoBldSrv triggers the script and returns result to the system



Subscription Service Design Concept

Script Management

1. EDEX client creates a subscription management request
2. Subscription request is passed via HTTP to the subscription manager
3. Subscription manager accesses /updates the subscription tables
4. Subscription manager prepares and returns a response to the EDEX client



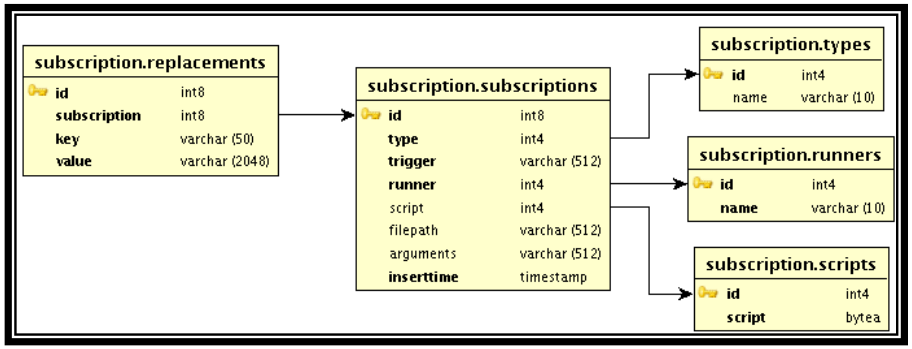
Subscription Service Design Concept

Additional Considerations

- Subscription management will be provided via the uEngine CLI tool
  - provides a simple command line interface
  - may be supplemented via Python/TK or another GUI interface
- Initially, we expect to provide 3 µEngine script runners
  - a Python script runner based on the existing Python µEngine
    - the actual script will be stored in the database
  - a JavaScript script runner based on the existing JavaScript µEngine
    - the actual script will be stored in the database
  - a system script runner based roughly on the adapter service
    - the idea is to allow running external scripts as if from command line

Subscription Service Data Structure

- Subscription information is stored in the database.
- The diagram represents the table structure that is envisioned for subscription management
- Two of the tables are generally static
  - the **types** table defines the available subscription trigger types (manual, timer, data)
  - the **runners** table defines the available uEngine types (python, jscript, system)
  - defining *types* and *runners* in the database allows flexibility and preserves data integrity

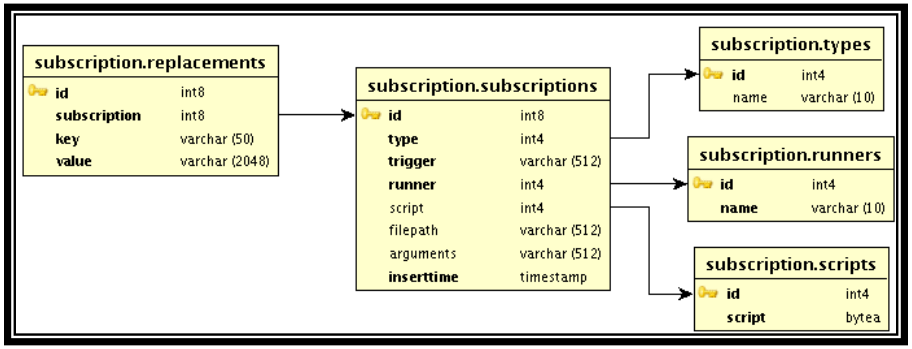


Have client read file

and send script to server.

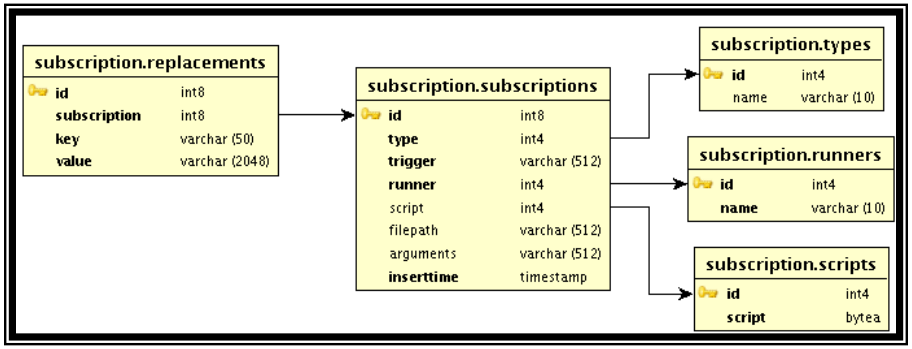
Subscription Service Data Structure

- The **scripts** table allows the same script to be used by multiple subscriptions.
  - e.g., a product generation script that creates a web displayable image might be used with multiple data arrival triggers
- The **replacements** table allows for replacement of *key tokens* by *value strings* prior to executing a script.
- A standard token of %TRIGGER% will be available to every *data* subscription
  - It will be replaced by the actual trigger.



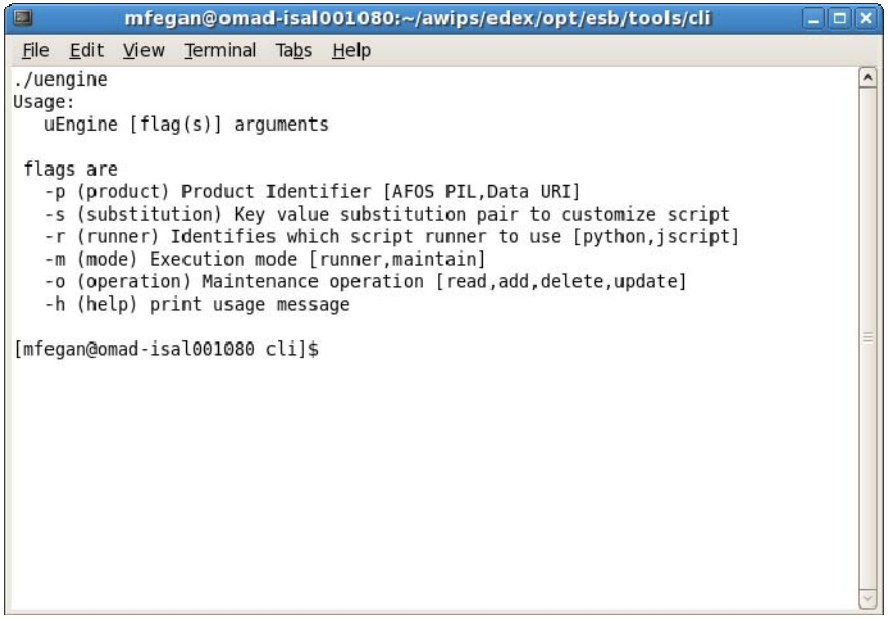
Subscription Service Data Structure

- The each row of the **subscription** table contains the information on a single subscription
- the *trigger* field identifies the trigger condition for fulfilling the subscription
  - for *timer* based subscriptions, it will be a chron expression
  - for *data* based dataURI expression
  - In both cases, the trigger may include wildcards to improve matching.
  - Note that the actual dataURI will be used for product retrievals



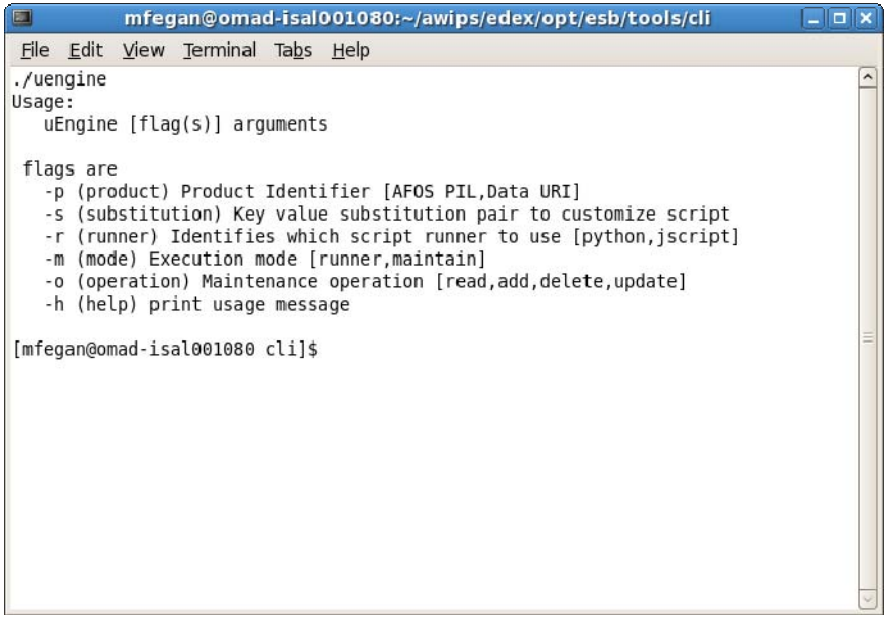
Subscription Service Interface

- Administrator interface to the Subscription Service is provided by the Micro Engine (µEngine) Command Line Interface (CLI) tool
  - subscription management is available by specifying the *maintain* mode.
- It may be desirable to provide an additional thin wrapper for uengine to script management; for example
  - **uengine** could be modified to automatically provide the *–m runner* argument
  - **subscription** could be created; it would automatically provide the *–m maintain* argument



Subscription Service Interface

- Some modifications to the uengine CLI tool will be required
  - additional options will be provided to allow specification of additional arguments:
    - trigger type (-t)
    - script file name (-f)
    - script file command line (-c)
  - uengine currently does not allow specifying both –r (runner) and –o (operation) flags
- Basic command line options are described on the next slide



Subscription Service Interface

Flag	Trigger	Runner	Description
-t	N/A	all	Identifies the type of trigger (timer, data, manual)
-r	all	N/A	Identifies the µEngine to execute the script (python, jscript, system)
-o	all	all	Identifies the operation (read, add, delete, update). Also implies maintenance mode.
-p	all	all	Identifies the pattern the trigger data must match to trigger subscription fulfillment. (Should be a P5 regex)
-s	all	p/j	Provides a key value pair identifying a substitution to be performed within the script prior to execution.
-f	all	s	Identifies the external script to execute. Must be a valid path to the script.
-c	all	s	Identifies command line argument(s) to pass to the script.

In addition to the command line arguments, scripts to be executed using either python or jscript µEngine script runners are provided via standard input.

Subscription Service Interface

- As a rather trivial example, we wish to add a Python script the logs “Hello from Omaha” into the EDEX log every hour (on the hour)
- Assuming the client script is in “data/HelloWorld.py” and that the script allows substitution for *MESSAGE*, the command line to create the subscription is

./uengine –m maintain -r python -o add -t timer
-p "0 .+ .+ .+ .+" -s MESSAGE "Hello from Omaha"
< data/HelloWorld.py

- If *subscription* is created, this becomes

./subscription -r python -o add -t timer
-p "0 .+ .+ .+ .+" -s MESSAGE "Hello from Omaha"
< data/HelloWorld.py

Subscription Service Interface

- A few more examples (assuming *subscription* CLI)
  - the command line to request a listing of all subscriptions is

    ./subscription -o read

  - the command line to request all subscriptions scheduled to run a 12:05 PM is

    ./ subscription -o read -p “5 12 .+ .+ .+”

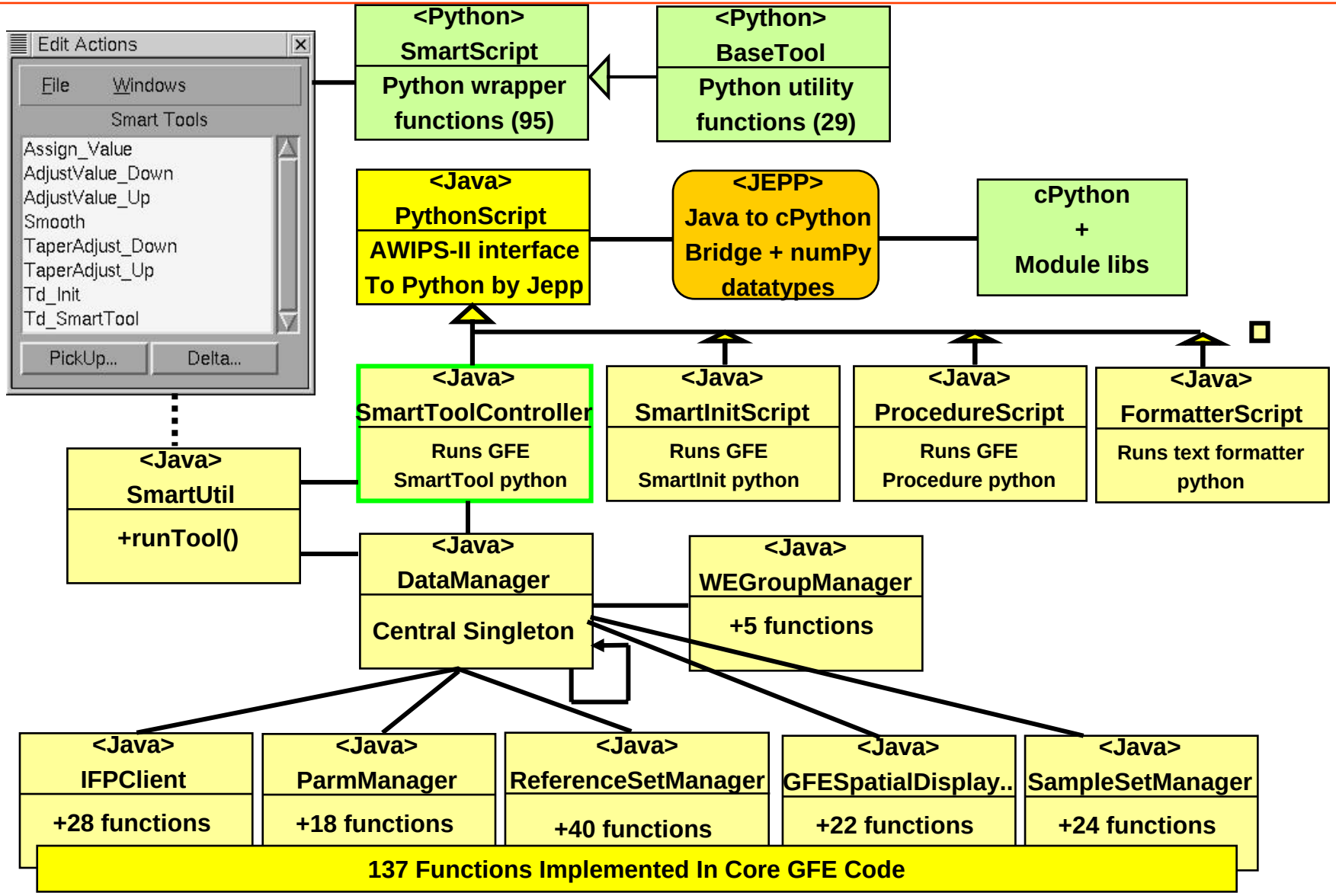
  - the command line to request a listing of all subscriptions for METAR observations is

    ./ subscription -o read -p "/obs/.+/METAR/.+"

Smart Tool Architecture Layers

- **JEP – Java Embedded Python**
  - Modified for improved performance and numpy support
- **PythonScript – Java class that wraps JEP calls**
- **SmartToolController (new in TO10)**
  - Java class that extends PythonScript
  - Manages smart tools in a single python interpreter (enables dropped in utilities to be picked up)
- **SmartToolInterface** – python class that assists the SmartToolController with managing and running smart tools
- **Tool** – Java class ported from Tool.py that prepares and executes smart tools through the SmartToolController
- **AFPS -** is incompatible with AWIPS-II. Most the methods used in AFPS, have a equivalent Java / Python method that can do the same thing. Translating a former AFPS call is straightforward, any that are missing should be brought to our attention.
- New Java class in T010 **"GFEProcedure"** runs procedures
- TO10 - **callSmartTool** and **callProcedure** added to SmartScript.py

AWIPS-II T010 Smart Tool Architecture Overview



AWIPS-II T010 Smart Tool Architecture

IFPClient functions (design at this level similar to A1)

Smart Tools

• Need to add method (i.e. getData(…)) to get at plug data other than GFE to replace direct access to netcdf.

IFPClient.java Public Functions

getParmList(DatabaseID id)
getGridParmInfo(List<ParmID> parmIds)
getSiteID()
getSiteTimeZone()
getGridInventory(List<ParmID> parmIds)
prepD2DData(ParmID parmId, TimeRange tr)
ge
re
ge
ge
ge
sa
ge
ge
ge
getReferenceData(ArrayList<ReferenceID>)
getGridData(ParmID parmId, List<TimeRange> gridTimes)
getParm(ParmID parmId, IParmManager parmMan)
commitGrid(List<CommitGridRequest> requests)
commitGrid(CommitGridRequest req)
getOfficialDBName()
getVTECActiveTable(String localSite)

SmartScript.py Function Available to SmartTools

```
def getGridLoc(self):
    if self.__gridLoc is None:
        parmIDs = self.__parmMgr.getAvailableParms(self.__mutableID)
        if len(parmIDs) > 0:
            tempParmID = parmIDs[0]
            info = self.__dataMgr.ifpClient().getGridParmInfo(tempParmID)
            self.__gridLoc = info.getGridLoc()
    return self.__gridLoc
```

Smart Tool Migration

- Imports from Numeric must be changed to import from numpy
- Wx tools must return a tuple of (numpy **byte** array, keys)
  - It appears that with numpy operations sometimes the array comes back as an array of type float32 (e.g. [0, 1, 0] could come back as [0.0, 1.0, 0.0])
  - If this occurs, at the bottom of the tool just ensure the wx is still of type byte
    - wxValues = asarray(wxValues, 'int8')
- Any calls on the C objects that did not go through SmartScript will have to be updated to call the correct Java methods

Analysis of RNK tools
Smart Tools

- RNK delivered Omaha 173 smart tools
- Analysis (all estimates)
  - 74 should work with TO9 with minor changes (previous slide)
  - 36 more should work with TO10 delivery with minor changes
  - 20 obsolete - Adjust tool makes others obsolete, some are point-based or parm-based tools
  - 30 incompatible – ISC, hardcoded directory paths, data type not currently supported in AWIPS II
  - Others unsure on compatibility – e.g. tkinter

Analysis of RNK tools
GFE Procedures

Smart Tools

- OB8.3 baselined procedures

```
CheckTandTd
CheckTTdWind
DiffFromClimo
GenerateCyclone
HazardRecovery
ISC_Discrepancies
MakeHazard          T010 Ported Warren
MergeHazards        T010 Ported Warren
PlotSPCWatches      T010 Ported Nate
PlotTPCEvents
PopulateFromClimo
PWS_Procedure
QPF_SnowAmt         T010 Ported Nate
SeparateHazards     T010 Ported Warren
TCMWindTool
ViewWCL
```

NCLADT TIM talking points/Questions

Elaborate on Raytheon responses to Govt. GFE questions

Govt. Comment: Each method of awips1's SmartScript.py gets used by some STR app at some site -- there is no SmartScript.py method that is not used. I see that some of awips1's SmartScript methods have been moved to Procedure.py in awips2

> **RTS Response:** The procedure methods have been moved back into SmartScript in TO10. Long story short, I was hoping to move from the GFE python style of everything inheriting everything, and move to a more modular design, but upon learning that there's so many custom procedures out there, that won't work when we're aiming for backwards compatibility. So those methods are back in SmartScript.

Govt. Comment: Utilities like the STR's ObjAnal, if dropped in the utilities directory, don't seem to get picked up by the system.

> **RTS Response:** I wasn't aware of that. The refactor of how smart tools work (single python interpreter) in TO10 should take care of this.

Govt. Comment: AFPS is not used only by SmartScript, but is called directly by many Tools and Procedures in the field.

> **RTS Response:** AFPS is incompatible with AWIPS II, no way around it. That said, for most of the methods used by AFPS, there is either an equivalent Java or Python method that can be used to accomplish the same thing. It's fairly straightforward, but if you find you need help translating a former AFPS call to the right call in AWIPS II, let me know.

NCLADT generated TIM talking points/Q

NCLADT

Govt. Comment: preProcessTool and postProcessTool need to be able to accept a variable length arg list

RTS Response: In TO10, the pre and post process methods work very similar to how they
worked in Tool.py. That is, they retrieve the method's argument names, and then use getArgValues() to get the argument's values.

Govt. Comment: How do procedures work? How is runProcedure.py run?

RTS Response: A new Java class in TO10 named GFEProcedure runs procedures. Running a procedure from the command line or a cron job (i.e. not CAVE), will not
work at this time.

Govt. Comment: callSmartTool and callProcedure don't exist in awips2's SmartScript.py

RTS Response: I've remedied that in TO10, successfully ported and tested.

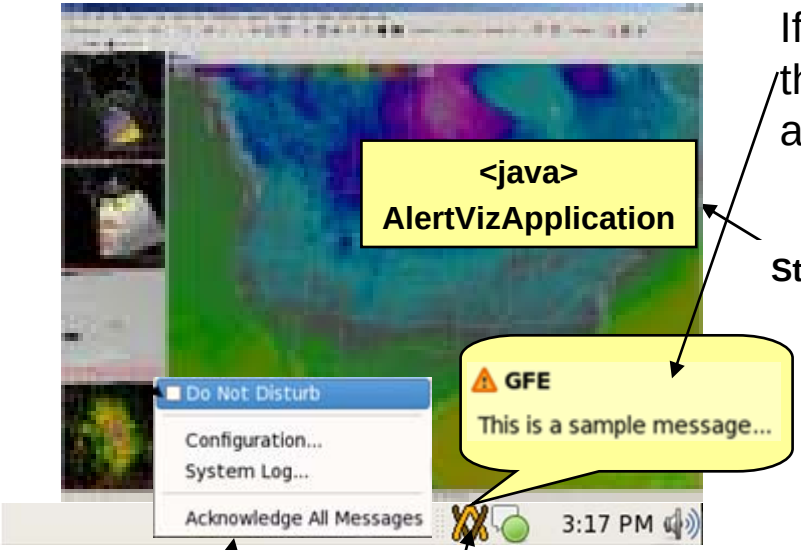
Govt. Comment: What is “grid access”

Govt. Comment: The ability to create, query, and archive gridded datasets outside of the default areas is vital. This piggybacks with the above concerns. It's ok if we don't use netcdf, but we need to have something to store to. For instance, ObjAnal reads in a series of observations points and interprets them to a uniform grid. This grid is then saved. This grid is used both to check forecast verification in the future, and improve the short term forecast in the operational

Guardian for Status and Logs

Centralizes status monitoring for all applications

Guardian



If the Layout (category) is set to system tray in the configuration dialog, incoming message will appear in a message balloon.

<java> AlertVizApplication

Stomp Message Protocal

<Python> sendNotificationMsg (command line utility)

Command line application allows you to send a message with a specified Category, Source, Priority, and message text.

Alert Visualization icon in the system tray.

Right-clicking the system tray icon will being up a popup menu.

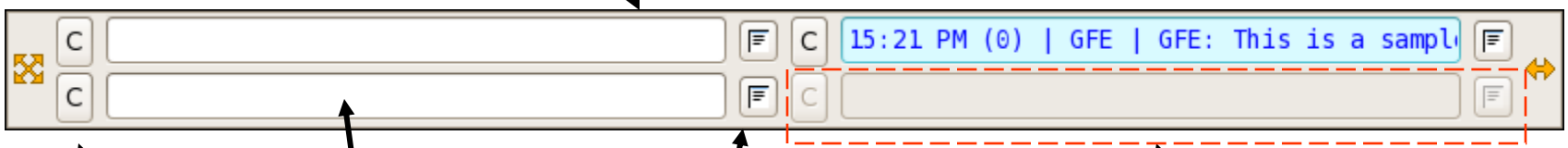
- **Do Not Disturb – Prevents the Alert Popup dialog from appearing. Any messages to be acknowledged will cause the system tray icon to flash.**
- **Configuration… - Shows the configuration dialog.**
- **System Log… - shows the System Log with all of the recent messages.**
- **Acknowledge All Messages – Acknowledge all of the Alert messages without bringing up the Alert Popup dialog.**

Guardian for Status and Logs

Alert Visualization dialog

Guardian

Alert Visualization dialog that displays the messages.



Clears Message

Displays Message

Shows Message Log

Message controls are disabled if there is not an assigned Category/Source.

Guardian for Status and Logs

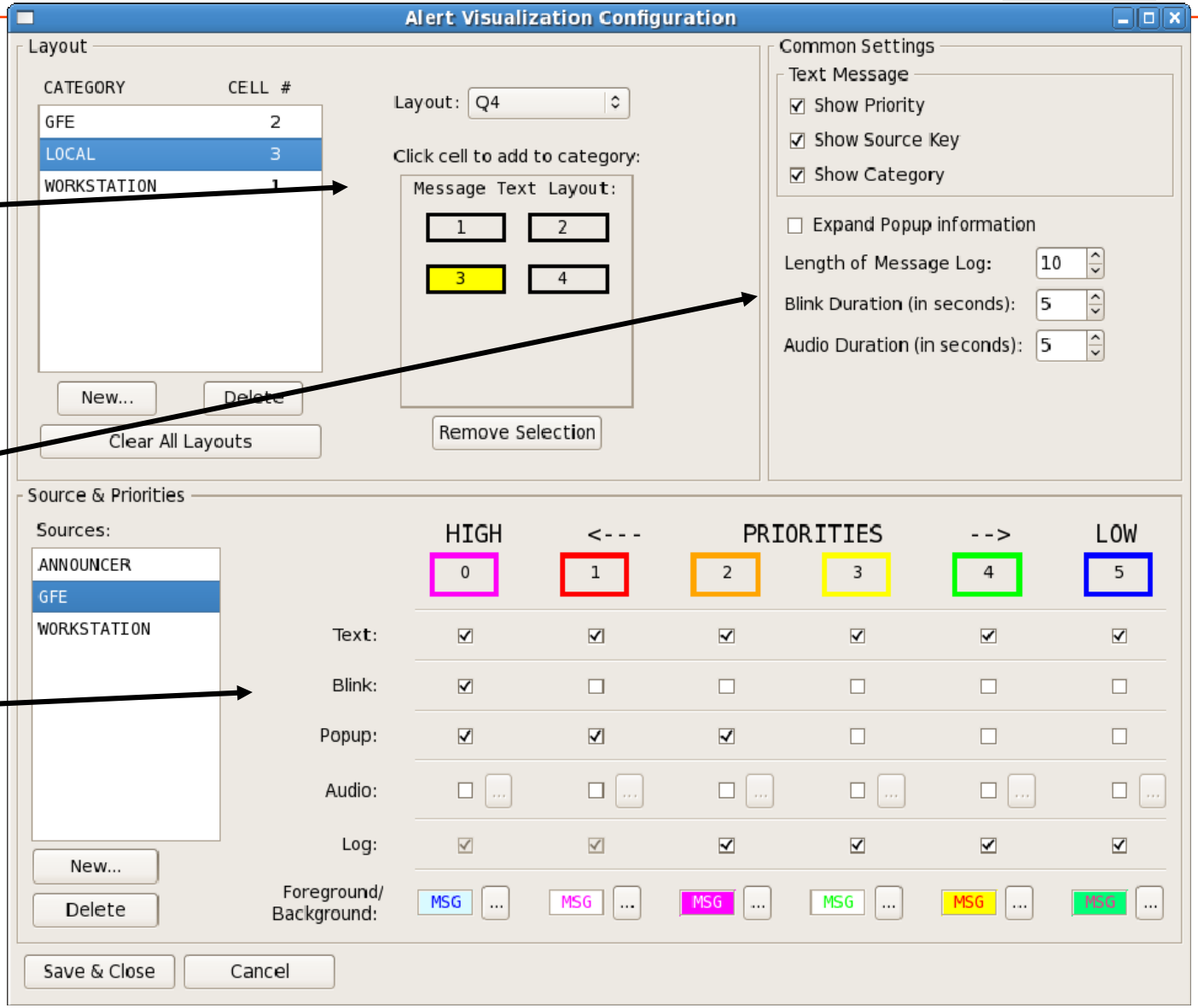
Configuration dialog

Guardian

Configure the category and where the message will be displayed in the Alert Message dialog (or can be put in the system tray).

Configure the common settings that apply to all messages.

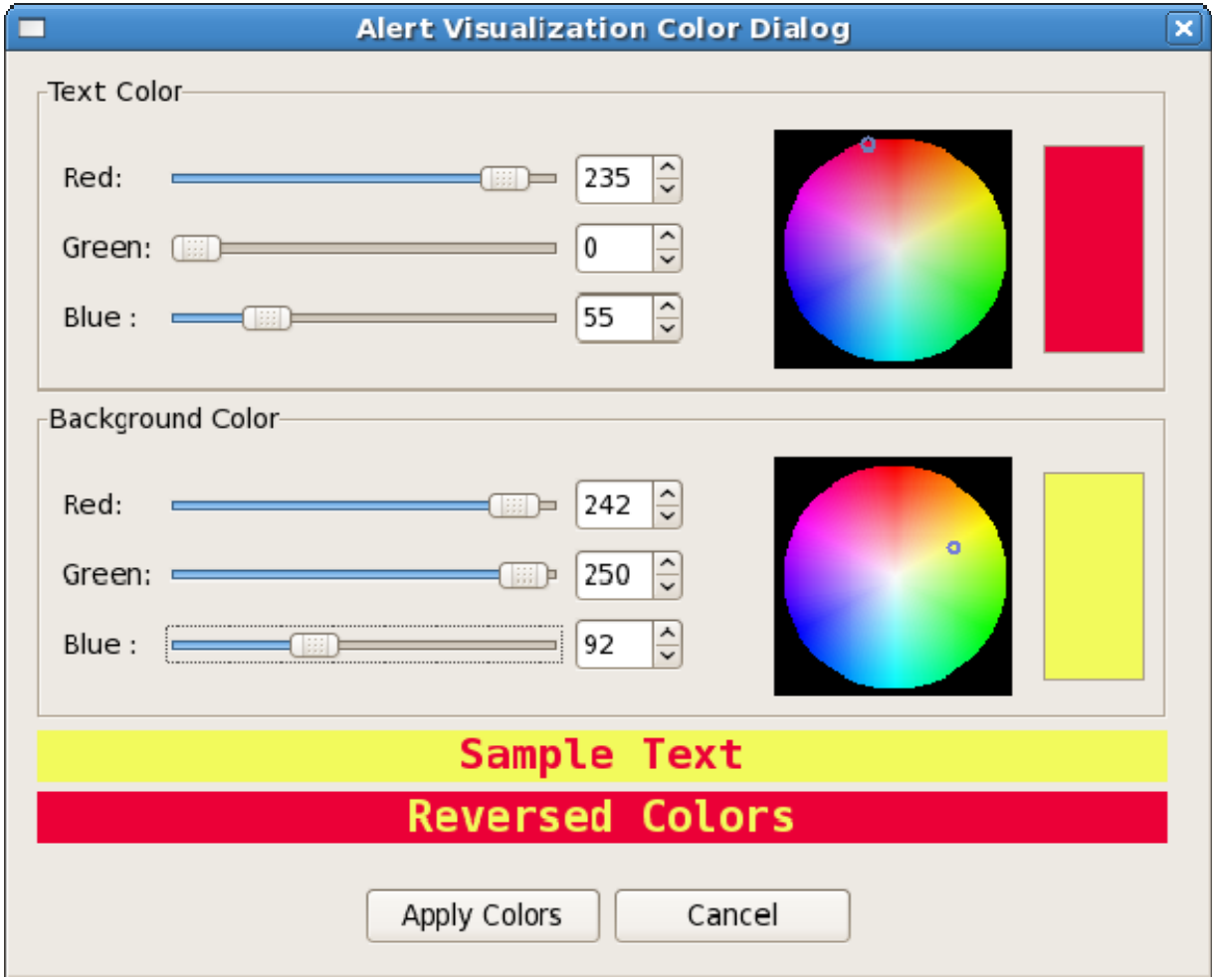
Configure how the message will be displayed for each source. This includes: blinking, audio alerts, and foreground & background colors.



Guardian for Status and Logs

Color dialog

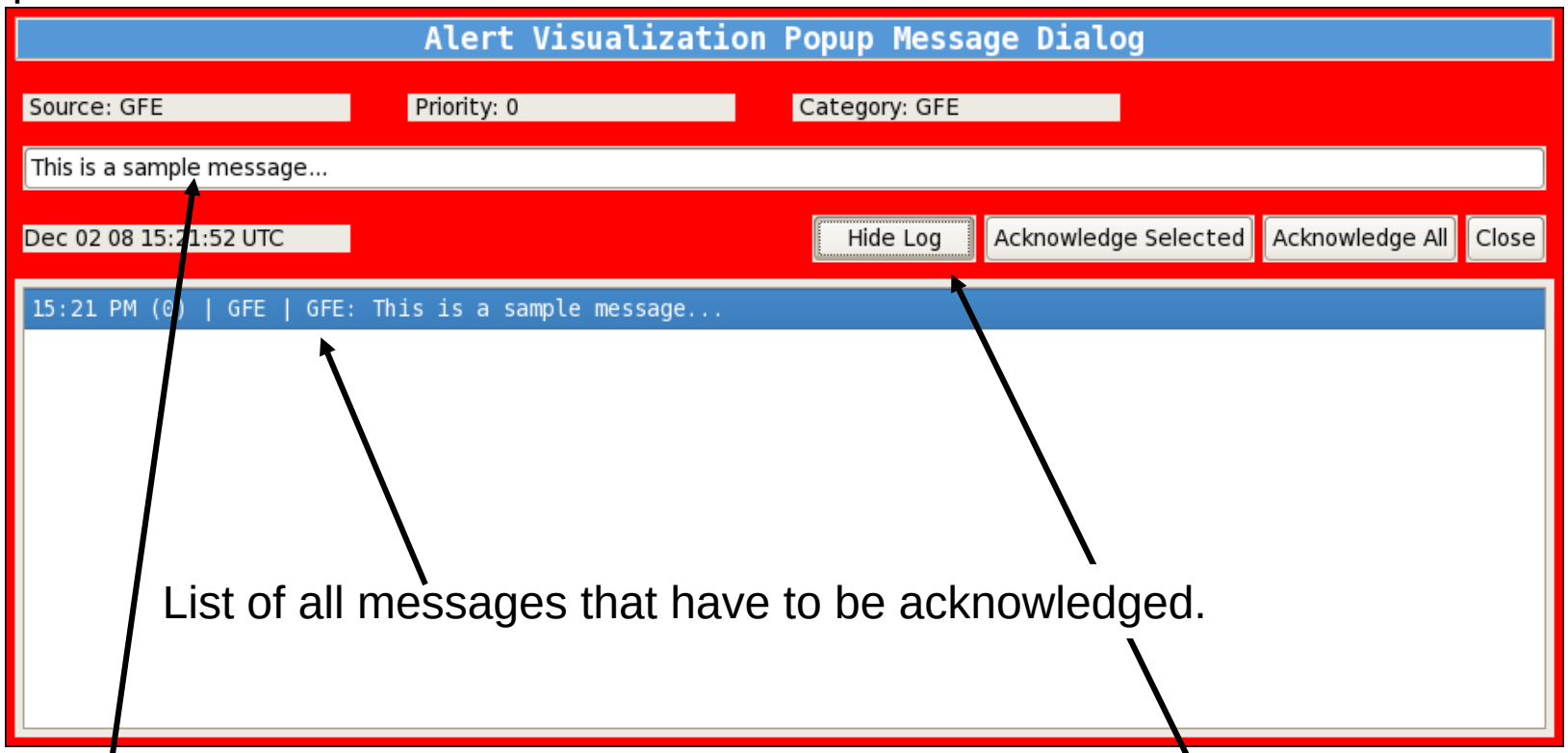
Color dialog used to change the foreground/background color of the messages on the Alert Visualization dialog. This is launched from the configuration dialog.



Guardian for Status and Logs

Alert Popup dialog

The Alert Popup dialog appears when the messages are configured to be a “popup” message. These messages have to be acknowledged by the operator.



List of all messages that have to be acknowledged.

Current message.

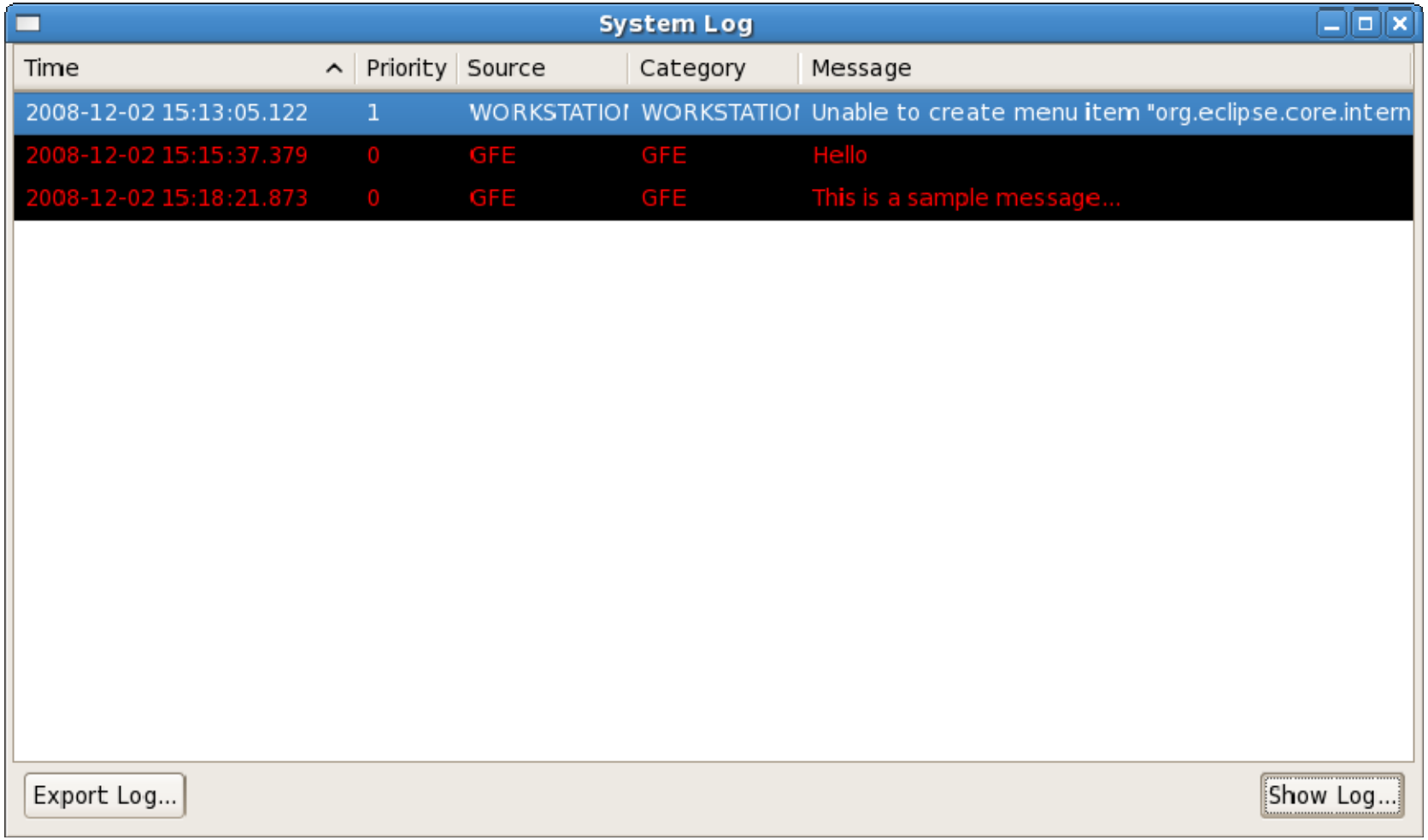
Show a list of all messages that have to be acknowledged.

Guardian for Status and Logs

System Log dialog

Guardian

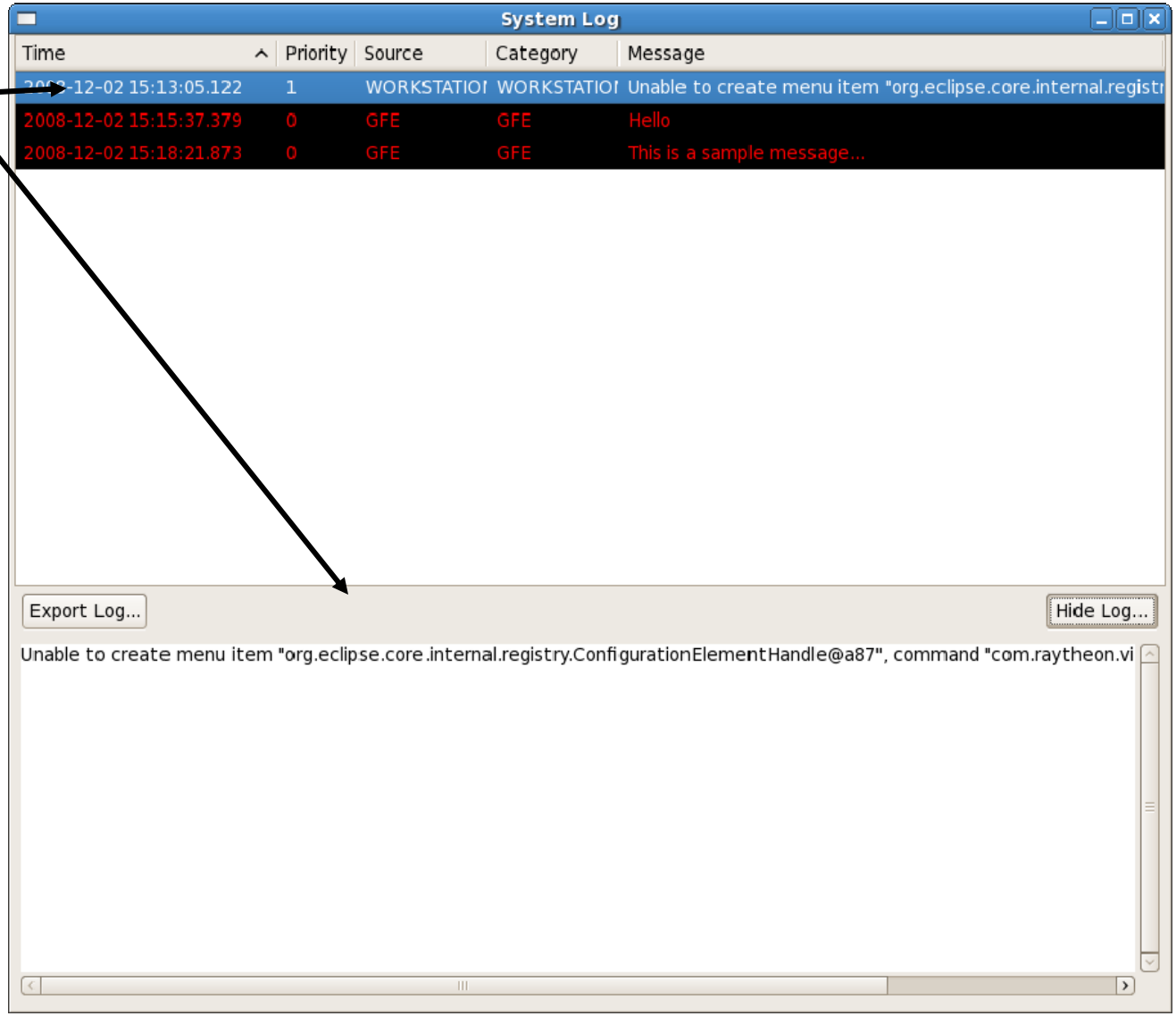
System Log dialog launched from the system tray icon popup menu. This dialog displays the list of messages that have been sent.



Guardian for Status and Logs

System Log dialog

Double-clicking a message will show the full log message with all details (like exceptions thrown).



NAWIPS Transition

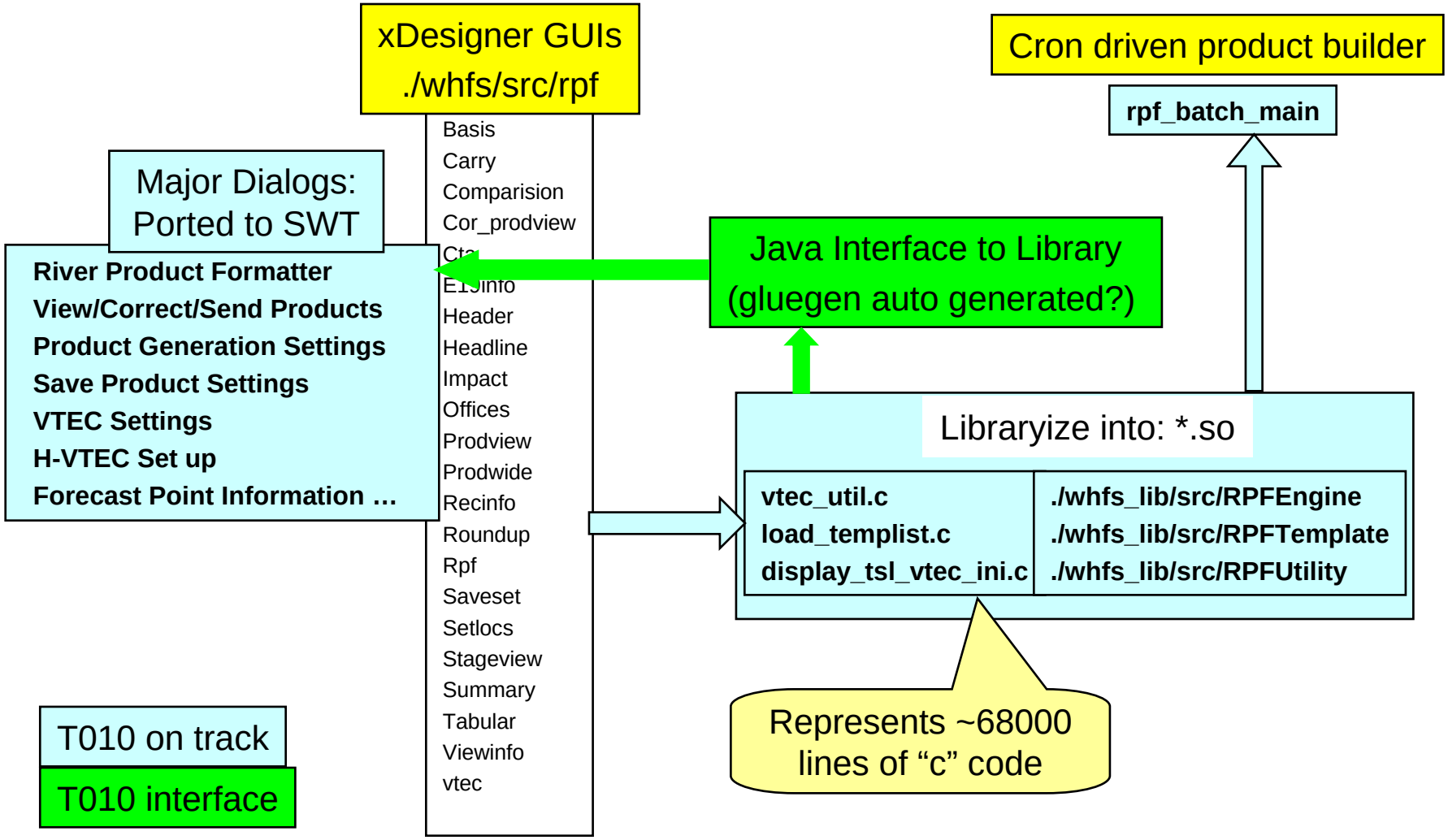
Discussion

AWIPS-II T010 RiverPro: Reengineering

KAP: ./awips/system engineering/T010/kap/RiverPro_Raytheon.ppt

Raytheon

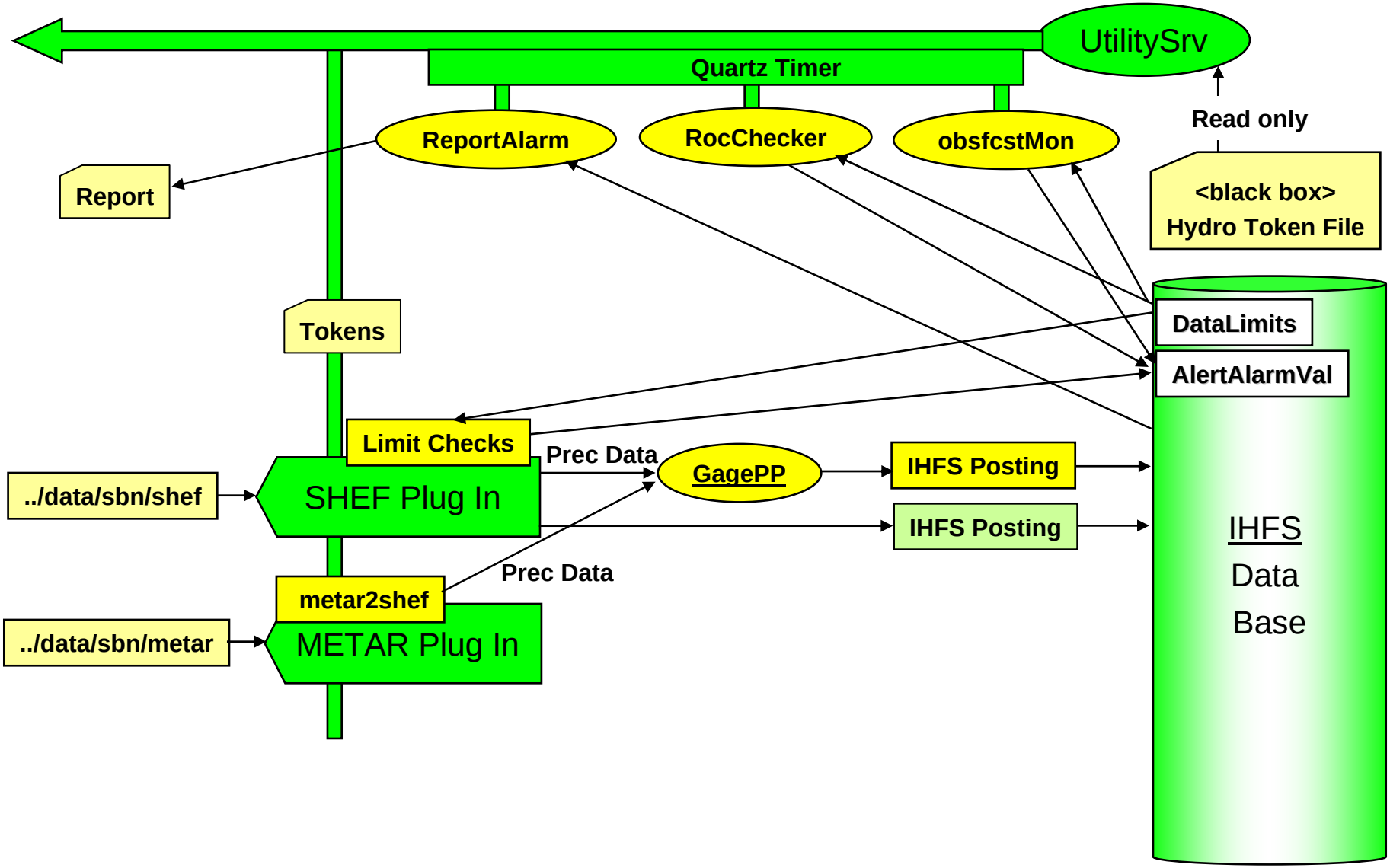
Hydro



T010 Hydro Ingest Processing

Gage Precip Ingest & Alert Alarm Generation

Hydro



Q&A