Raytheon

Customer Success Is Our Mission

Advanced Weather Interactive Processing System II (AWIPS II)

AWIPS Development Environment (ADE) and the Common AWIPS Visualization Environment (CAVE)

Introduction

February 14, 2008

AWP.TRG.SWCTR/TO8.ADE/CAVE-00.00

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 the restriction are contained in all sheets.

Purpose of Course

- Provide an early developer-level introduction to facilitate cooperative development
  - Trying to evolve project toward an Open Source core
  - Everything a part of the baseline and open to improvement
- Provide up-to-date information on ADE 1.0 (TO 8) Updates
  - Modified EDEX Ingest Data Flow
  - Data Access Layer Design Modifications
  - Database Implementation Modifications
  - Modifications to AWIPS II Installer

Training Prerequisites

- Reading Materials
  - *Software Product Improvement Plan*
- Software
  - Pure Java
    - http://java.sun.com/docs/books/tutorial
  - CAVE: ECLIPSE IDE Framework & Plug-Ins
    - http://www.eclipse.org
    - Eclipse RCP
  - EDEX: Introductory level of Spring and Mule ESB
    - http://mule.codehaus.org
  - All: Introductory level of ANT
  - All: Introductory level of XML
- AWIPS II Training Modules 1 – 11 (as revised for TO8)
- NWS Training – AWIPS II FAQ, ADE 1.0 Addendum

Course Objectives

Module 12: TO8 ADE 1.0 Developer Updates

- Describe revised Ingest Data Flow
- Describe revised Data Access Layer implementation
- Describe revised Database definition pattern
- Review modifications to the Plug-In creation utility
- Review Installer modifications for TO8

Note: *Modules 1 – 11 were previously delivered and are not included in this course. Updates for TO8 are to be delivered by the end of February.*

Raytheon

Customer Success Is Our Mission

Advanced Weather Interactive Processing System II (AWIPS II)

AWIPS Development Environment (ADE) and the Common AWIPS Visualization Environment (CAVE)

Module 12: TO8 ADE 1.0 Developer Updates

February 14, 2008

AWP.TRG.SWCTR/TO8.ADE/CAVE12.00

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 the restriction are contained in all sheets.

Objectives

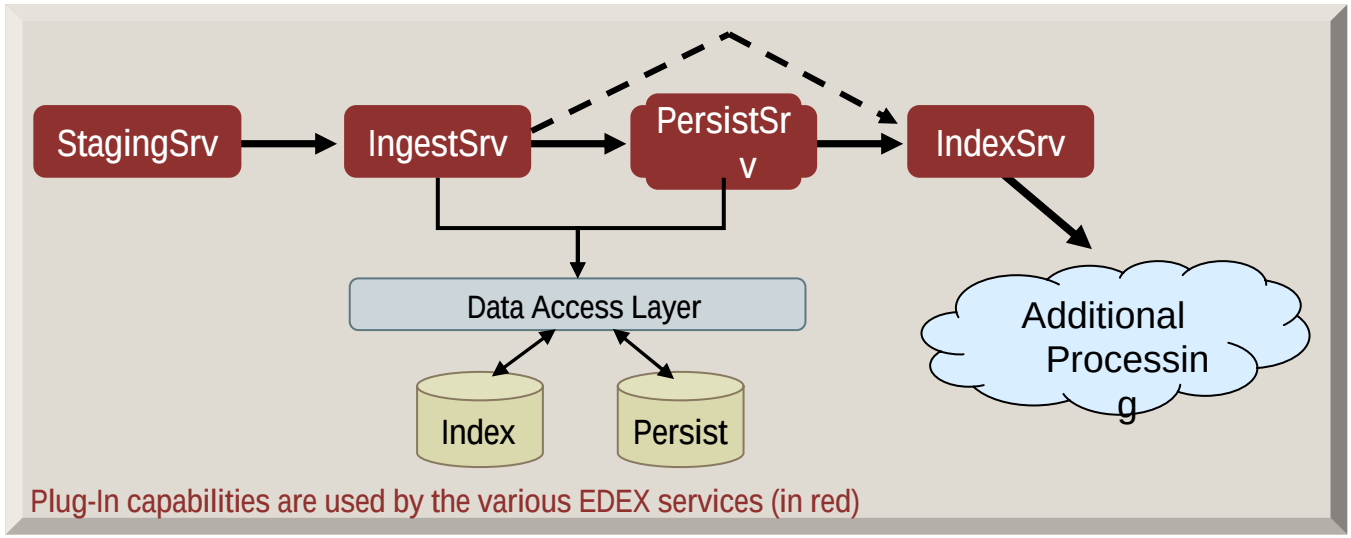
- Provide updates to previously covered developer topics including
  - Modification to EDEX Ingest data flow and impact to data-type plug-in development
  - Modification to EDEX Data Access Layer and impact to data-type plug-in development
  - Modification to EDEX Data Base Definition pattern and impact to data-type plug-in development
  - Modifications to the EDEX Plug-in Creation Tool
  - Additions to the AWIPS II installers
  - Preview of upcoming training and capabilities

Revised Ingest Data Flow

Modified EDEX Ingest Data Flow

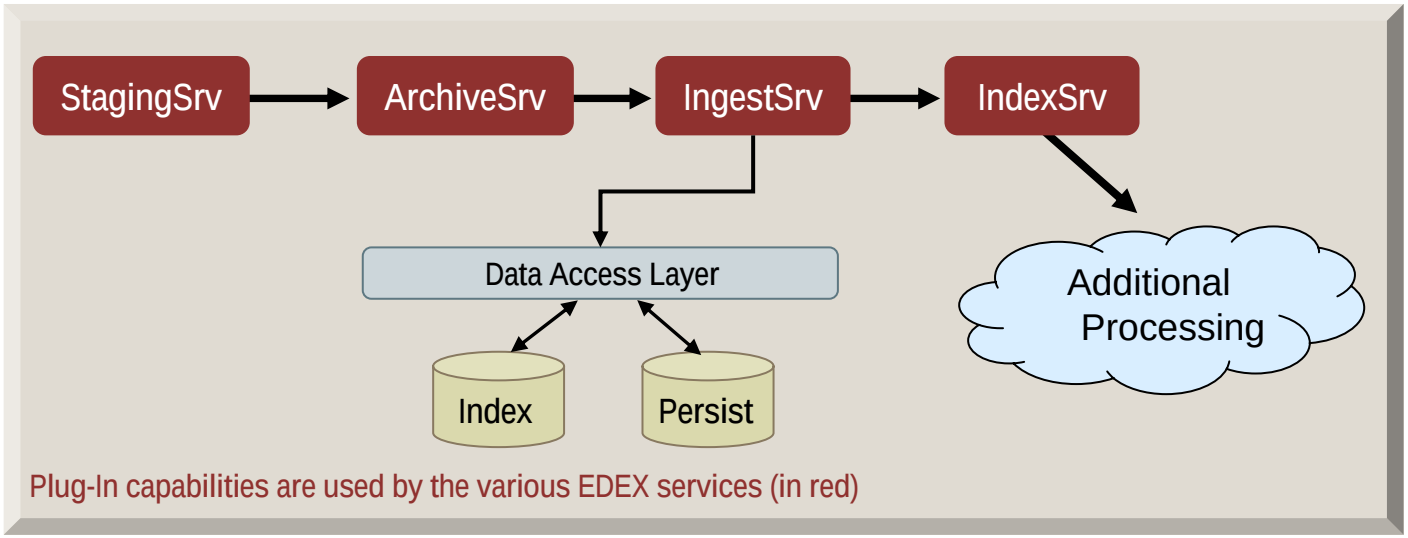
- Existing Ingest Data Flow was causing memory-related problems
  - This is critical to system stability in the future.
- For TO8, EDEX Ingest DATA Flow is simplified and standardized
- Standardized: All data initially handled by a staging service to facilitate automated load balancing in a clustered environment
- Standardized: All data passes through an archive service to allow archiving of data for future playback
- Simplified: Ingest and persistence services combined into a single service
- Simplified: Standardization of data flows simplifies the configuration of data endpoints

Ingest Data Flow Delivered in TO 6



- Features separate mule endpoints for:
  - Reading data files (StagingSrv)
  - Decoding the data (IngestSrv)
  - Saving Data to HDF-5 (PersistSrv)
  - Writing meta-data to the database (IndexSrv)
- Not all data processed by PersistSrv

Ingest Data Flow Delivered in TO 8



- New service (ArchiveSrv) added to provide archiving of inbound data
- IngestSrv now handles both decoding of data and persistence to HDF-5 based data store
- StagingSrv, ArchiveSrv and IngestSrv have a separate end-point for each data type

TO 8 EDEX Ingest Services

Service	Description
StagingSrv	Listens on an SOA endpoint for data and moves the data to working directory on a network shared device. Initially places file path on JMS queue to allow cluster-based processing.
ArchiveSvr	Listens to JMS queue for available work. Copies files from working directory to archive directory. Returns file path to JMS queue for further processing.
IngestSrv	Listens to JMS queue for available work. Reads file from working directory and decodes file contents. Persists certain binary data to HDF-5 archive. Passes metadata to JMS queue for further processing.
IndexSrv	Listens to JMS queue for available work. Saves the metadata extracted from the ingested data to PostgreSQL database for client retrievals. Passes meta-data to JMS queue for further processing.

Data Flow Impact

- First three stops of the modified data flow configured in a single XML file in each data-type plug-in
  - XML file is the xxx-ingest.xml file, located in the “res/endpoints” directory of the plug-in (xxx represents the data type, e.g. satellite)

    ***[Note**: See next few slides for the default configuration for the satellite plug-in.]*
- Final stop of the data flow configured in the Index Server’s configuration file, index.xml
  - This file is located in “opt/esb/conf”
- All configuration files located in the EDEX baseline in the ADE

Demo: Use Eclipse file browser to show location.

satelite-ingest.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mule-configuration PUBLIC "-//SymphonySoft //DTD mule-configuration XML V1.0//EN"
   "http://www.symphonysoft.com/dtds/mule/mule-spring-configuration.dtd">

<mule-configuration version="1.0">
   <model name="edex" type="seda">

   <!-- Endpoint to stage Satellite data -->

   <!-- Endpoint to archive Satellite data -->

   <!-- Endpoint to ingest Satellite data -->

   </model>
</mule-configuration>
```

Add configuration for endpoints.

This is the basic SOA configuration file. Content represented by the three comments is shown on the next three slides.

satelite-ingest.xml (cont'd)

```
<!-- Endpoint to stage Satellite data -->
<mule-descriptor
     name="Awips.Edex.Service.StagingSrv-satellite"
     singleton="true"
     implementation="com.raytheon.edex.services.StagingSrv
     outboundEndpoint="jms://ar/sat">
   <inbound-router>
      <endpoint name="sbnSatIngestEndpoint“
                address="file://../../data/sbn/sat/?transformers=NoActionTransformer">
         <properties>
            <property name="moveToDirectory“
                      value="../../processing" />
         </properties>
      </endpoint>
   </inbound-router>
   <threading-profile maxThreadsActive="1" maxThreadsIdle="1" />
</mule-descriptor>
```

EDEX Class handling message.

Outbound JMS queue.

Inbound source directory.

Move to working directory.

Single instance used for this end point.

This is configuration for the satellite staging service. It is configured to "sniff" a directory and move each file to the working directory.

satelite-ingest.xml (cont'd)

```
<!-- Endpoint to archive Satellite Data -->
<mule-descriptor
     name="Awips.Edex.Service.ArchiveSrv-satellite"
     singleton="false"
     implementation="com.raytheon.edex.services.ArchiveSrv"
     outboundEndpoint="jms://cp/sat">
   <inbound-router>
      <endpoint name="AR-Sat" address="jms://ar/sat" />
   </inbound-router>
   <threading-profile maxThreadsActive="4" maxThreadsIdle="4" />
   <properties>
      <property name="pluginName" value="SATELLITE" />
      <property name="archiveDirectoryLocation"
                value="../../data/archive/sat/" />
      <property name="jmxModeOn" value="true" />
   </properties>
</mule-descriptor>
```

EDEX Class handling message.

Outbound JMS queue.

Inbound JMS queue.

Up to 4 instances used for this end point.

Move to working directory.

This is the configuration for the satellite archive service. It is configured to listen on a JMS queue and process the file identified by the message received from that queue.

satelite-ingest.xml (cont'd)

```
<!-- Endpoint to ingest Satellite data -->
<mule-descriptor
     name="Awips.Edex.Service.IngestSrv-satellite"
     singleton="false"
     implementation="com.raytheon.edex.services.IngestSrv">
   <inbound-router>
      <endpoint name="CP-Sat" address="jms://cp/sat" />
   </inbound-router>
   <outbound-router>
      <router className="org.mule.routing.outbound.FilteringListMessageSplitter">
         <endpoint address="vm://indexVMQueue" />
      </router>
   </outbound-router>
   <threading-profile maxThreadsActive="1" maxThreadsIdle="1" />
   <properties>
      <property name="pluginName" value="SATELLITE" />
   </properties>
</mule-descriptor>
```

EDEX Class handling message.

Inbound JMS queue.

Outbound JMS queue, splits multiple messages.

Thread pooling for this endpoint.

This is the configuration for the satellite ingest endpoint. It is configured to listen on a JMS queue and process the file identified by the message received from that queue.

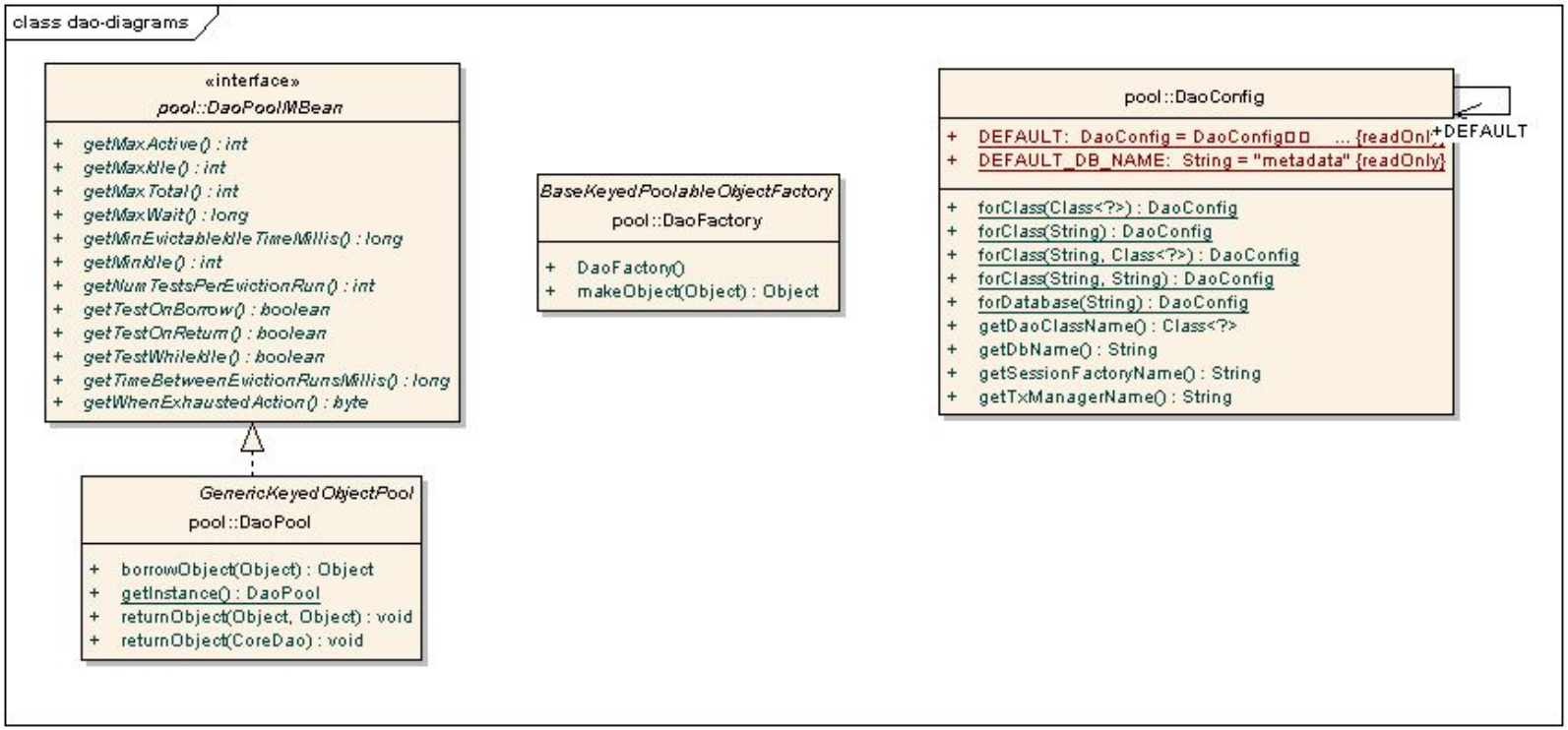
Questions?

Revised Data Access Layer Implementation

DAO Pooling: Overview

- Database performance, specifically slow database access times, identified as an issue coming out of TO6
  - The Data Access Layer (DAL) was instantiating and destroying a Data Access Object (DAO) for each database interaction
  - Because the DAO encapsulated the database connection, this tends to be a (time-) expensive operation
- Pooling mechanism introduced in TO8 to limit the potentially expensive operation of instantiating data access objects
  - DAO created the first time it is needed, then maintained in a pool of available objects
  - When a client needs to access the database, it gets the DAO from the pool rather than creating a new DAO object instance

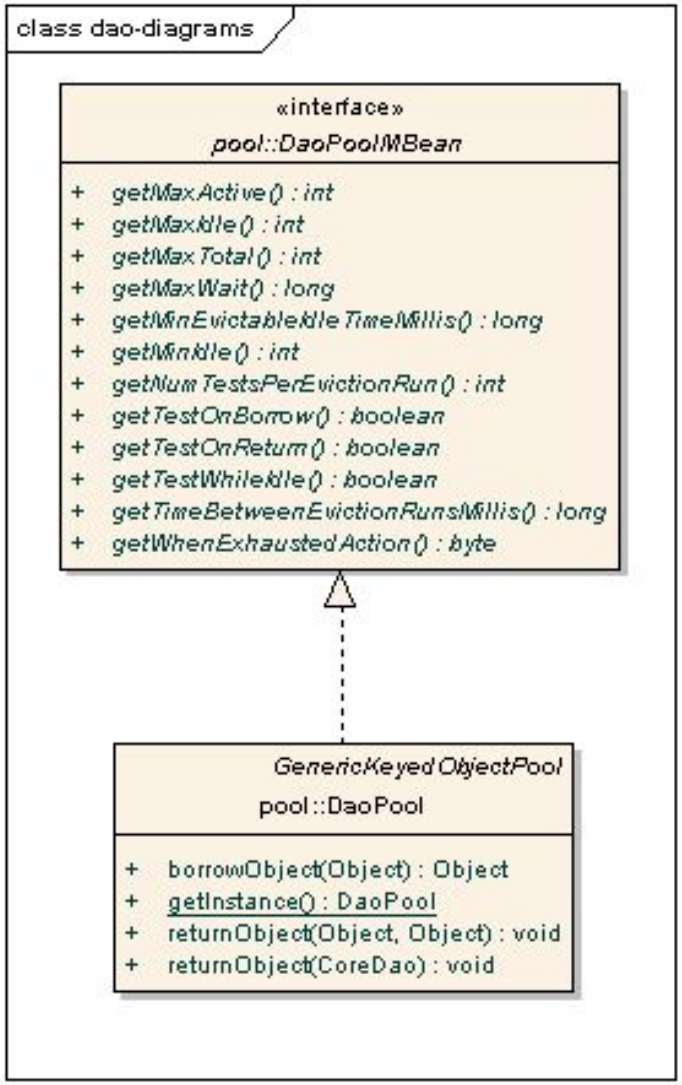
DAO Pooling: Overview (cont'd)



- DAO pooling mechanism consists of three classes
  - DaoPool: A singleton, this is the class clients utilize to borrow and return a DAO
  - DaoFactory: The class is used internally by DaoPool to instantiate appropriate DAO
  - DaoConfig: This class contains information about the DAO; the database name, Hibernate class, and the Hibernate session factory and transaction factory. Provides static convenience methods for obtaining configurations.

DAO Pooling: DaoPool Class

- Main class used for borrowing data access objects
- Consists of two main methods:
  - borrowObject(). Used to obtain the DAO from the pool
  - returnObject(). Used to return the DAO to the pool
- Every call of borrowObject() must be followed by a matching call of returnObject()



DAO Pooling: Borrowing a DAO

- Make a call to DaoPool.getInstance().borrowObject(Object key);
- Key may be any of 3 types object types: String, Class<?> and DaoConfig
  - String object: The value of the String is the name of a database
    - A data access object for the specified database is returned
    - The DAO will be usable only for database inserts and updates (Hibernate requires a class instance to retrieve data)
  - Class<?> object: The Class object of a DAO
    - An instance of the DAO is returned
    - The DAO has full database functionality available
  - DaoConfig object: An instance of DaoConfig describing the desired DAO
    - The DaoFactory is used to construct a DAO matching the properties of the DaoConfig object.
    - This option allows user to create a DOA “on the fly” when the DAO does not already exist

DAO Pooling: Utilization

This code snippet is from the AIREP Decoder.

- As with any pooled resource, one must be careful to return borrowed objects to the pool.

```
AirepDao dao = (AirepDao) DaoPool.getInstance()
               .borrowObject(AirepDao.class);
try {
   // set up dataURI
   Object [] dbRpt = dao.queryByDataURI(dataURI);
   // use the results of the query
} catch (Exception e) {
   // Handle Exception
} finally {
   DaoPool.getInstance().returnObject(dao);
}
```

- Always use a **try – finally** block when using the DAO pool
- As shown here, the DAO class object is normally used to borrow the DAO instance

DAO Pooling: Pool Configuration

- Pool Configuration options located in the DALConfig.xml file
- These may be changed, but will not take effect until Mule is restarted.

```xml
<property name="props">
   <props>
      <prop key="maxActive">150</prop>
      <prop key="maxIdle">50</prop>
      <prop key="maxTotal">200</prop>
      <prop key="maxWait">-1</prop>
      <prop key="minEvictableIdleTimeMillis">10000</prop>
      <prop key="minIdle">0</prop>
      <prop key="numTestsPerEvictionRun">10</prop>
      <prop key="testOnBorrow">false</prop>
      <prop key="testOnReturn">false</prop>
      <prop key="testWhileIdle">false</prop>
      <prop key="timeBetweenEvictionRunsMillis">60000</prop>
   </props>
</property>
```

DAO Pooling: Additional Information

- EDEX Database pooling utilizes the generic object pooling API provided by the Apache Commons Pool project
- More information available at: http://commons.apache.org/pool/

Questions?

Revised Database Definition Pattern

EDEX Plug-Ins: The Plug-In Concept

- EDEX plug-in contains multiple classes and definition files
  - A sample plug-in directory structure – the satellite decoder plug-in – shown here
  - A plug-in is not limited to this directory structure; it may contain additional directories
- EDEX services use the Plug-In to decode data, store data, retrieve data and convert data for transfer to clients

```
plugin-satellite
|-- res
|    |-- conf
|    `-- endpoints
|-- src
|    `-- com
|         `-- raytheon
|              `-- edex
|                   |-- plugin
|                   |    `-- satellite
|                   |-- units
|                   |    `-- satellite
|                   |         |-- goes
|                   |         |    `-- convert
|                   |         |-- ir
|                   |         `-- water
|                   `-- util
|                        `-- satellite
```

Note: "com/raytheon" may be replaced by "gov/lab"

EDEX Plug-Ins: Plug-In Configuration Files

- Most aspects of an EDEX plug-in controlled by a set of files
  - Although not strictly required, most Plug-Ins have these files

Folder	File	Description
<root>	build-components.properties	Defines the build dependencies for the plug-in
	client-includes.dat	Defines the files to include in the client version of the plug-in
<root>/res	binding.xml	Defines the JiBX bindings for data classes in the plug-in
	<data-type>.db.xml	Defines the database table definitions for the plug-in
	<data-type.hbm.xml	Defines the Hibernate mappings for the plug-in
<root>/res/conf	plugin.xml	Defines run-time configuration values for the plug-in
<root>/res/endpoints	<name>-ingest.xml	Defines the Ingest ESB definitions for using the plug-in for data ingest

Note:

- <name> is the plugin name, e.g., grib
- <root> is plugin-<name>, e.g., plugin-grib

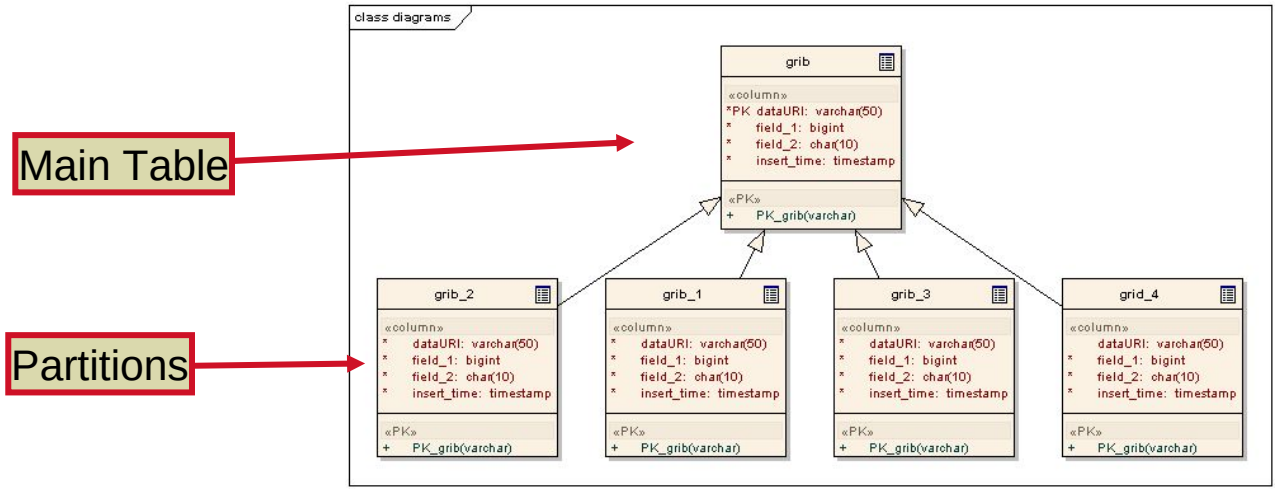
EDEX Plug-Ins: DB Table Generation

- EDEX has ability to generate Plug-In related database tables automatically
  - Table generation controlled by the Plug-In's <data-type>.db.xml files
  - Each file contains the definition of a single database file
- A Plug-In's tables are generated when the Plug-In is first added to the EDEX system
  - Tables are also added following a complete database purge

EDEX Plug-Ins: Data Retention

- EDEX uses the data table definition file to specify data retention for automatic data management
  - Maintenance consists of deleting old data files (HDF5) and purging old data from the database (PostgreSQL)
  - Data maintenance is performed
    - When the EDEX server initially starts
    - At 15 minutes past the hour as EDEX continues to execute
- EDEX uses a partitioned approach to data management
  - Data partitioned into 4 segments
    - In the database, each data table has 4 partitions
  - Data deletion accomplished by deleting/purging appropriate partition
- Example: If a retention time of 24 hours is specified, each partition will contain 12 hours of data
  - More on this later

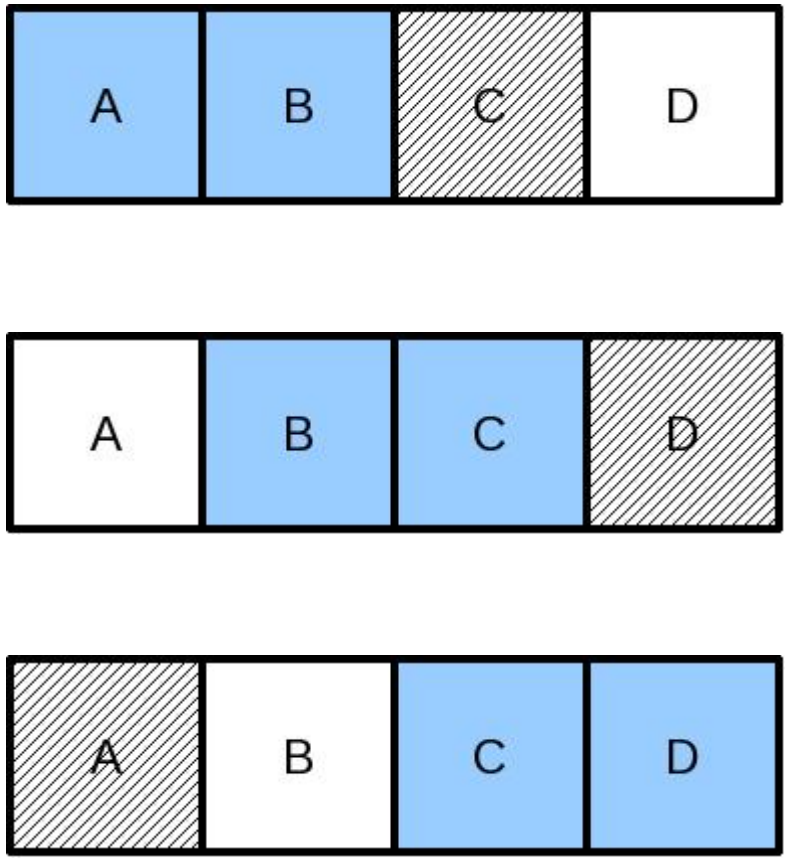
EDEX Plug-Ins: Data Table Partitions



- EDEX database uses PostgreSQL partitioning of database tables
  - Each table has 4 partitions
  - Each time data is purged; the oldest partition is deleted and re-created
- Partition usage (once the server is running)
  - Two partitions contain archived data
  - One partition is being actively used to store new data
  - One partition is being deleted and re-created

EDEX Plug-In: Data Table Partition Example

- Assuming a specified retention time of 24 hours
  - Each partition will hold 12 hours of data
- Data stored into partition A & B, D as a "spare"
  - After 24 hours, data begins to store into C
  - A minimum of 24 hours data is retained
- When C is filled, data is stored into D
  - At that point, A is purged and re-created
  - Once again, a minimum of 24 hours
- When D is filled, data is stored into A
  - At that point, B is purged and re-created
- This process continues as long as EDEX continues to execute!



EDEX Plug-Ins: DB Table Definition

```
<tableDefinition>
    <tableName>grib</tableName>
    <hibClass>com.raytheon.edex.plugin.grib.GribRecord</hibClass>
    <retentionHours>24</retentionHours>
    <order>1</order>
    <defaultClass>true</defaultClass>
    <partitioned>true</partitioned>
    <linkedExternally>true</linkedExternally>

    <columnDefinition>
        <name>datauri</name>
        <columnType>varchar</columnType>
        <constraintType>PRIMARY KEY</constraintType>
        <precision>512</precision>
        <scale>0</scale>
        <dataURI>false</dataURI>
        <index>none</index>
    </columnDefinition>

     ...

</tableDefinition>
```

Example of a db.xml file. The next few slides explain each element.

EDEX Plug-Ins: DB Table Definition (cont'd)

Tag	Description
tableDefinition	Defines a single table. Each table definition file contains a single tableDefinition tag set
tableName	Defines the name of the database table to create
hibClass	Defines the name of the class that represents a record in this table
retentionHours	Defines the number of hours to retain data in this table
order	Defines the order in which related tables are created
defaultClass	Specifies if this table is the default (base) table for the plug-in
partitioned	Specifies whether the table may be partitioned
linkedExternally	Specifies whether entries in this table are linked to data in the HDF5 repository
columnDefinition	Defines a column in the table. A each table contains multiple column definitions

EDEX Plug-Ins: DB Table Definition – Additional Information on Table Definition

Raytheon

- *order* tag: A Plug-In may specify multiple tables, each specification in a separate file. The order tag determines the order in which tables are created. Each table in a Plug-In must have a different order value. This allows a Plug-In to:
  - Specify multiple tables and
  - Include dependencies between the tables.
- *defaultClass* tag: A Plug-In may define multiple tables for its data. The defaultClass tag specifies if the table is the main for the Plug-In. This table is the one used by term query. Exactly one table may be designated as the default.

EDEX Plug-Ins: DB Column Definition

Tag	Description
columnDefinition	Defines a single column of the table
name	The name of the column
columnType	The column type
constraintType	Constraint type for the column. Valid values are PRIMARY KEY, UNIQUE, FOREIGN KEY, NONE
precision	The size, number of characters or digits, of the field
scale	(For numeric floats only) the number of decimal places
dataURI	Specifies whether this column is used in creating the data DUI
index	Specifies whether this column is indexed. Valid values are single, composite, none

EDEX Plug-Ins: DB Column Definition – Data URI Column

Raytheon

- Main table for a data-type plug-in includes a data URI column
  - The data URI column is the primary key for this table.
  - For externally linked tables, the data URI column links the meta-data in the table to the binary data in the HDF5 repository.
  - Selected columns in the table contribute to the data URI. These columns must be selected to provide a unique column value.

```
<columnDefinition>
    <name>datauri</name>
    <columnType>varchar</columnType>
    <constraintType>PRIMARY KEY</constraintType>
    <precision>256</precision>
    <index>none</index>
</columnDefinition>
```

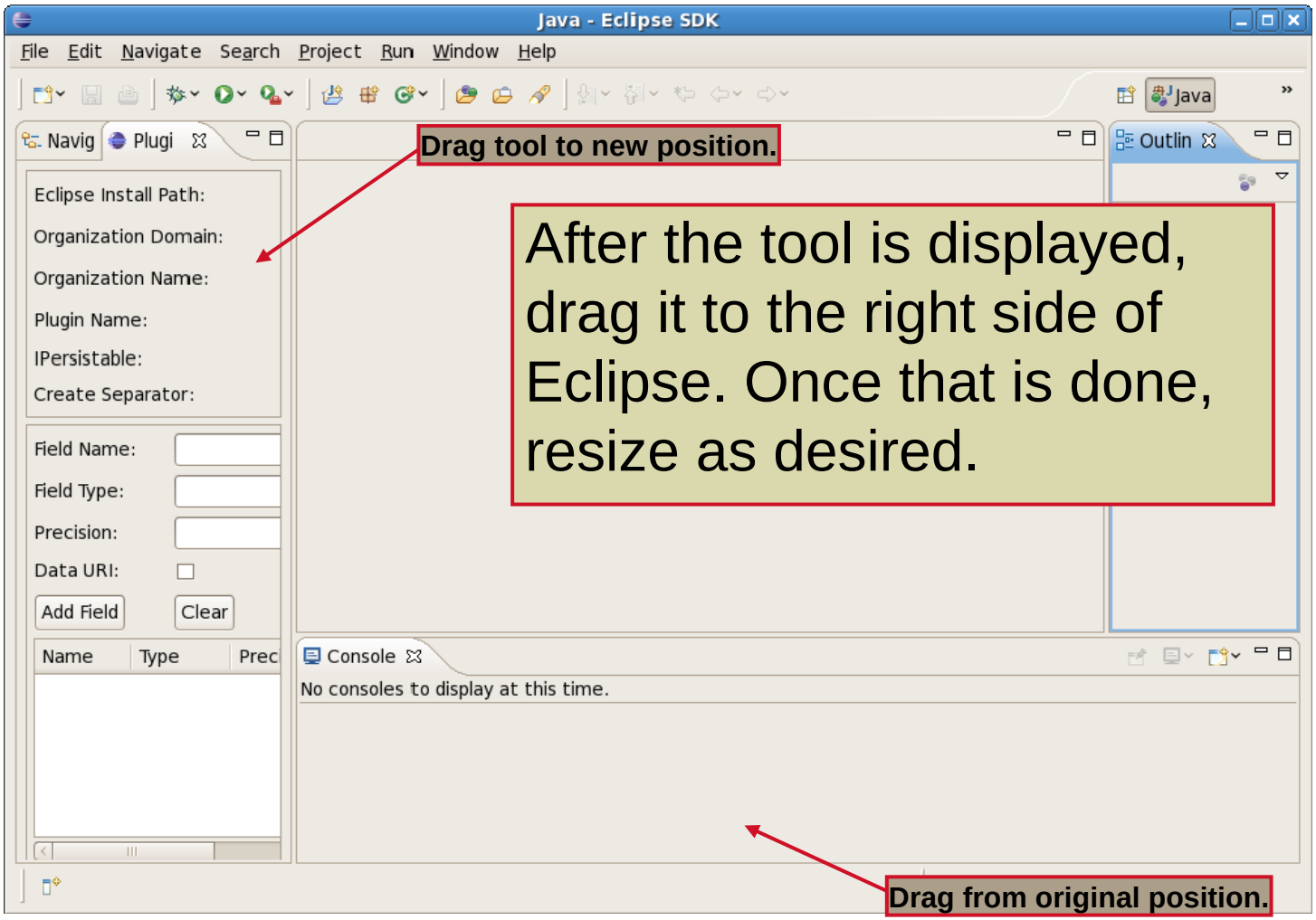
Questions?

Tool Modifications for TO8

Problems Identified/Addressed

- Problems with the original Plug-In Tool identified by NWS:
  1. A need to better handle the “package name space”
  2. A need for scroll bars to be able to use the entire tool
- In addition, there was some refactoring of the Plug-In pattern
  3. A need update the tool to match the refactored Plug-In
- Of these,
  - #1 and #3 addressed via code changes
  - #2 is a consequence of Eclipse. We present a partial workaround. At this point, there is no way to use the tool on a “small” screen.

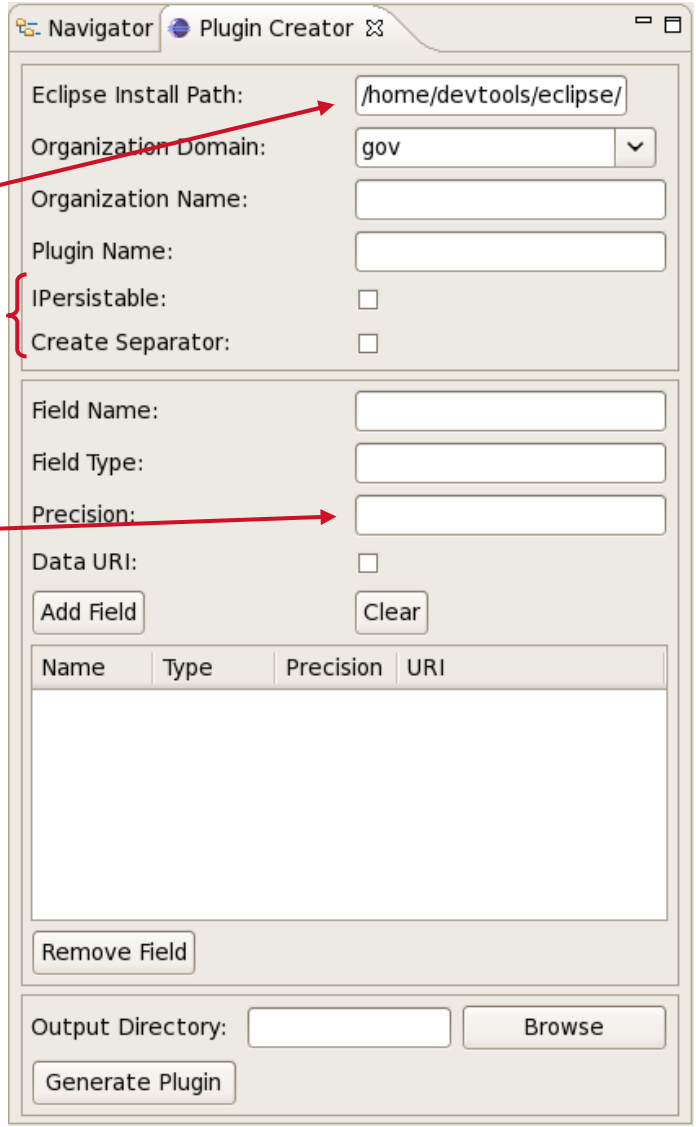
Displaying Plug-In Tool



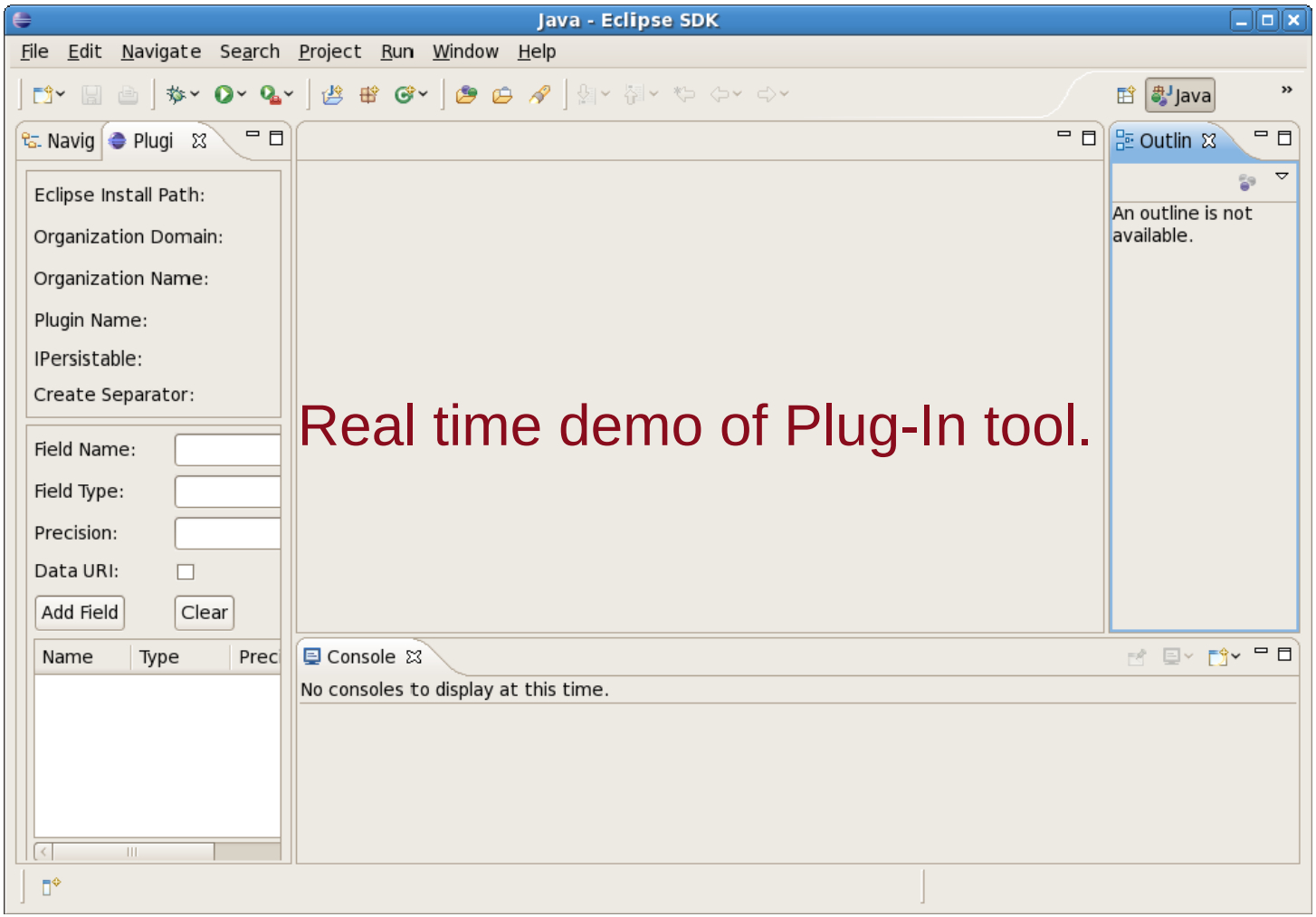
After the tool is displayed, drag it to the right side of Eclipse. Once that is done, resize as desired.

Plug-In Tool, New Features

- The Plug-In tool has been modified slightly to better fit thePlug-In creation pattern.
  - Code templates are part of the Plug-In, so you need to specify your Eclipse location.
  - Wording of checkbox labels has changed – does not change functionality.
  - A precision value has been added to the section for defining data entries. This is used mainly for defining database fields for strings. It should be set to zero (0) for other data types.



Plug-In Tool, Demonstration



Real time demo of Plug-In tool.

Questions?

Code Example

Code Walk-Through

Examination of Code for a Data-Type Plug-In Using Eclipse

Installer Modifications for TO8

AWIPS II Installer

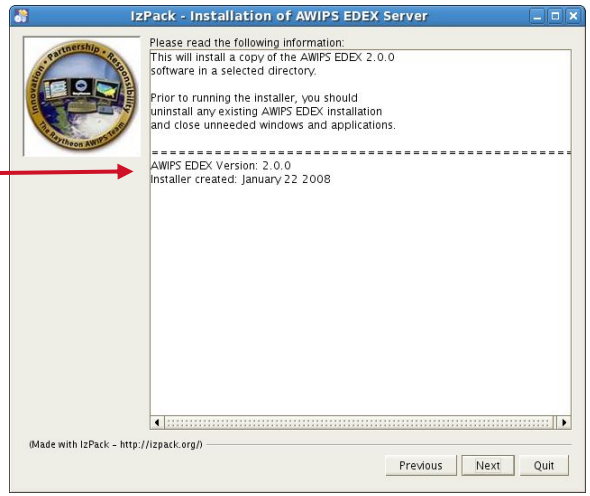
- With AWIPS II, there are now 3 installers
  - CAVE installer
  - EDEX installer
  - ADE installer

- AWIPS II Installers require Java

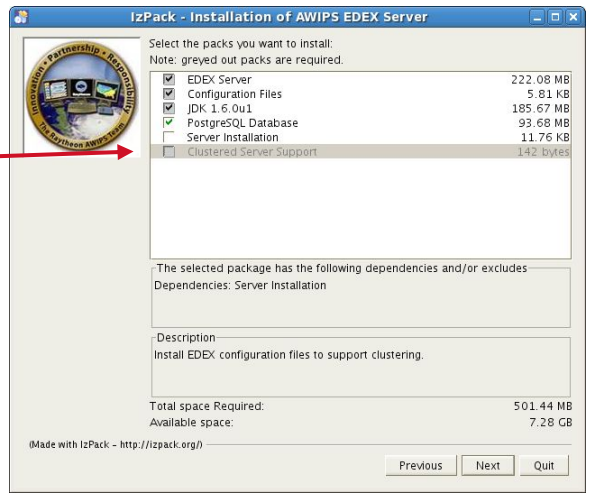
Note: *Support for EDEX installer on Windows has been dropped with T08.*

AWIPS II Installer Updates

- Versions have been updated
  - Shown here is the general information page, also on other pages

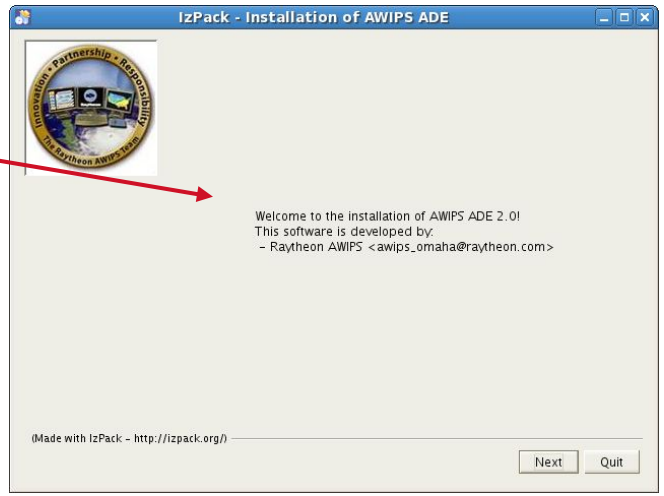


- Pack selection page has
  - Fewer choices
  - Support for clustered servers

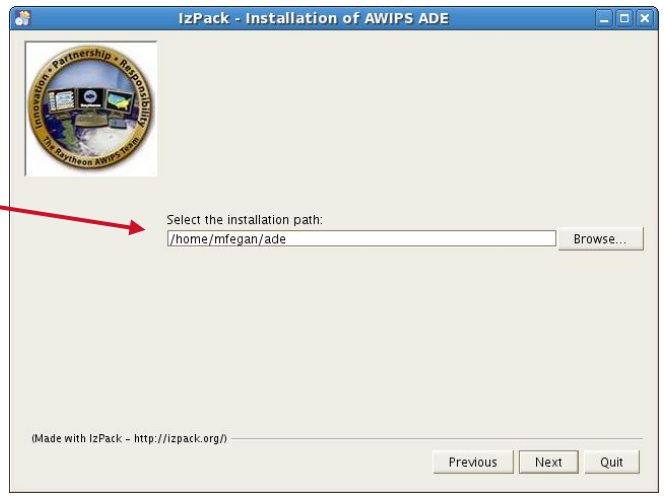


ADE Installer

- AWIPS II ADE 2.0 now has a separate installer
- Installs development tools
  - AWIPS II (TO 8) code base
  - Eclipse Europa (3.3.0)

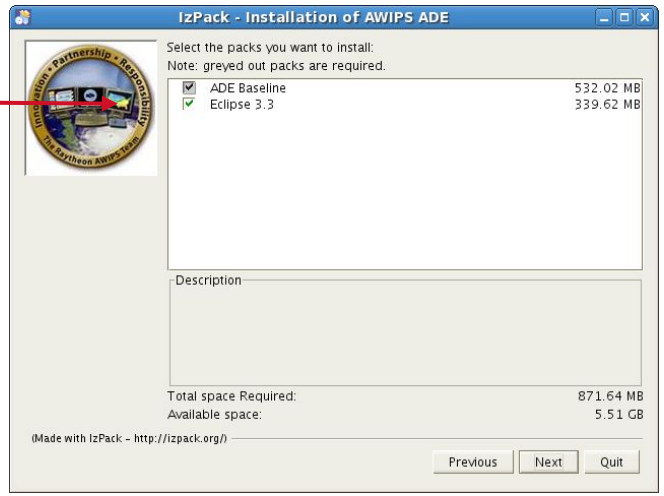


- ADE installation location is selectable at install time
  - Default location is “~/ade”



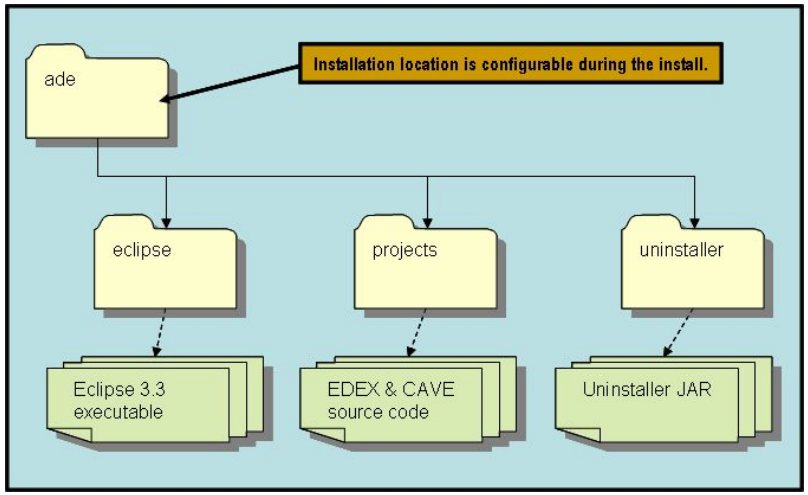
ADE Installer (cont'd)

- ADE components installed are selectable
  - Must install code base
  - Eclipse installation is optional
    - Eclipse install required for CAVE work



- Uninstaller provided as part of the ADE installation

ADE Installer (cont'd)



- Basic ADE 2.0 install structure shown here
  - Name and location of the root directory is configurable (at installation)
  - Default installation location:

    <user-home>/ade (Linux)

    C:\Program Files\ade (Windows)

Questions?

CAVE Modifications for TO8

Other Items of Interest

CAVE Modifications for TO8

- TO8 implemented selected AWIPS I functionality in CAVE
  - For the developer, this amounts to an addition of code to the baseline
  - The basic patterns and techniques used by the CAVE developer have not changed – refer to existing AWIPS II modules
- Any questions?

Coming in TO9

- Additional AWIPS I functionality
  - Core GFE
  - AVN FPS
  - D2D functionality
- Python Scripting support
- Local application support

Coming in TO T1 Training Updates

- Documentation of existing uEngine Tasks
- Updates to existing Training Modules (1 – 11)
- Annotated code examples

Questions?

Wrap-Up

Summary

- Covered modifications in EDEX dataflow introduced in TO8
- Covered modifications to the Data Access Layer Implementation introduced in TO8
- Covered modifications to the Database Definition Pattern introduced in TO8
- Covered modifications to the to the Plug-in Creation Utility introduced in TO8
- Covered modifications to the Installer for TO8

Resources

- On the ADE 1.0 DVD (TO8)
  - Current code available for examination in the CAVE baseline
  - JavaDoc documentation available