Presenting the Proof-of-Concept NWS Workflow Management System

A first implementation of high-level recommendations for a future RFS architecture

September 3, 2003

APEX DIGITAL SYSTEMS

Agenda

- The Challenge: New Requirements demand a New Architecture
- Brief review of the high-level architectural priorities presented to OHD by Apex on June 13, 2003
- Overview and demonstration: The proof-of-concept NWS Workflow Management System
- Conclusions & next steps

The Challenge: New Requirements demand a New Architecture

- OHD is driving scientific advances and implementing new forecasting applications on a continuous basis
- Distributed modeling will make a major impact, both scientifically and functionally on OHD's work in the future
- OHD also faces ever-increasing functional requirements from its customers
- The existing system architecture presents significant limitations to continued effective implementation of new forecasting capabilities

Key Requirement: Any Candidate Architecture Must Continue to Support Existing Applications

- OHD must continue forecasting operations at all times. As a result, the implementation of a new architecture requires a gradual and evolutionary approach, and must support existing modes of operation
- The new architecture should be flexible and modular to improve on the current monolithic system environment

Brief Review of High-Level Architectural Priorities

In a meeting between OHD and Apex on June 13, 2003, Apex proposed several architectural priorities to focus and guide the evaluation process towards specific outcomes in improving NWSRFS:

A. Standardize data management and data delivery
B. Implement a flexible application configuration environment
C. Implement active application management
D. Encapsulate existing Fortran applications to operate within the application configuration and management environment

The proof-of-concept system presented here successfully addresses three of the four priorities

Priority A: Standardize data management and data delivery

- Current concerns regarding data management:
  - Data mostly stored in files, using a variety of data formats, some of which are binary
  - Data files are tightly coupled with applications
  - All data management is local to each AWIPS configuration at RFCs
- Proposed approach:
  - Store all data in relational databases
  - Use OLTP and OLAP architectures as appropriate
  - Implement a transparent framework for delivering data locally or remotely, i.e., loosely couple data into applications
  - Deliver data with platform and device independence using a newly adopted or designed Hydrologic Markup Language

Priority B: Implement a flexible application configuration environment

- Current concerns regarding application configuration:
  - Applications are either tightly coupled to each other with explicit launch calls or completely uncoupled and launched by CRON daemons
  - Applications use non-standard control files with typically unique formats and content based each application
  - Applications are compiled into one monolithic AWIPS environment, forcing the emergence of “in-official” configurations
- Proposed approach:
  - Use a managed environment based on an application server
  - Architect an environment that supports the structural and process differences between science development and production uses
  - Create an environment that enables and supports rapid, standards-based reconfiguration of application workflows
  - Maintain the power and simplicity of a meta-language such as HCL

Priority C: Implement Active Application Management

- Current concerns regarding application management:
  - Applications tend to be designed as single executables with less focus placed on integration into workflows
  - Workflow or queue management is "simulated" with CRON daemons
  - Logging and application monitoring is relatively unstructured
  - Performance is limited due to local program execution and whole-database locking by each session
- Proposed approach:
  - Design and implement applications as workflow components
  - Implement workflow or queue management that is location and device transparent
  - Implement structured application event logging and monitoring that is human and machine readable and can be used in workflow management
  - Enable distributed application execution

Priority D: Encapsulate existing Fortran applications for continued functionality

- Current concerns regarding Fortran components:
  - Fortran should increasingly be regarded as legacy code
  - Fortran is not directly supported in contemporary managed application frameworks
- Proposed approach:
  - Where possible, migrate applications to C or C++
  - In other cases, attempt to separate monolithic applications into smaller functional components, and wrap each component into a C or C++ wrapper
  - Wrap entire programs in C or C++ wrappers and redirect or reload program output to make it available in the managed environment

The Proof-of-Concept System Demonstrates the Architectural Priorities in Action

- The proof-of-concept NWS Workflow Management System successfully demonstrates 3 of the 4 priorities:
  - Priority B: Application configuration via XML
  - Priority C: Active application management
  - Priority D: Encapsulated Fortran code
- Improved data management (priority A) will require further discovery and design work to define the scope and impact of transitioning data management capabilities to more structured repositories

Overview and Demonstration: The Proof-of-Concept NWS Workflow Management System

- The proof-of-concept challenge
- Overview of system components
- General architectural overview
- Component detail
  - Workflow server
  - Workflow client
  - Messaging-enabled applications
  - Logging server
  - Message delivery framework
- Example Workflows

The Proof-of-Concept Challenge

To build a simple runtime environment that supports distributed, messaging-enabled applications that function in the context of managed workflows.

Some of the applications must encapsulate existing code, which must continue to function in a stand-alone fashion.

Further, the proof-of-concept must be a functioning system, not just a plan.

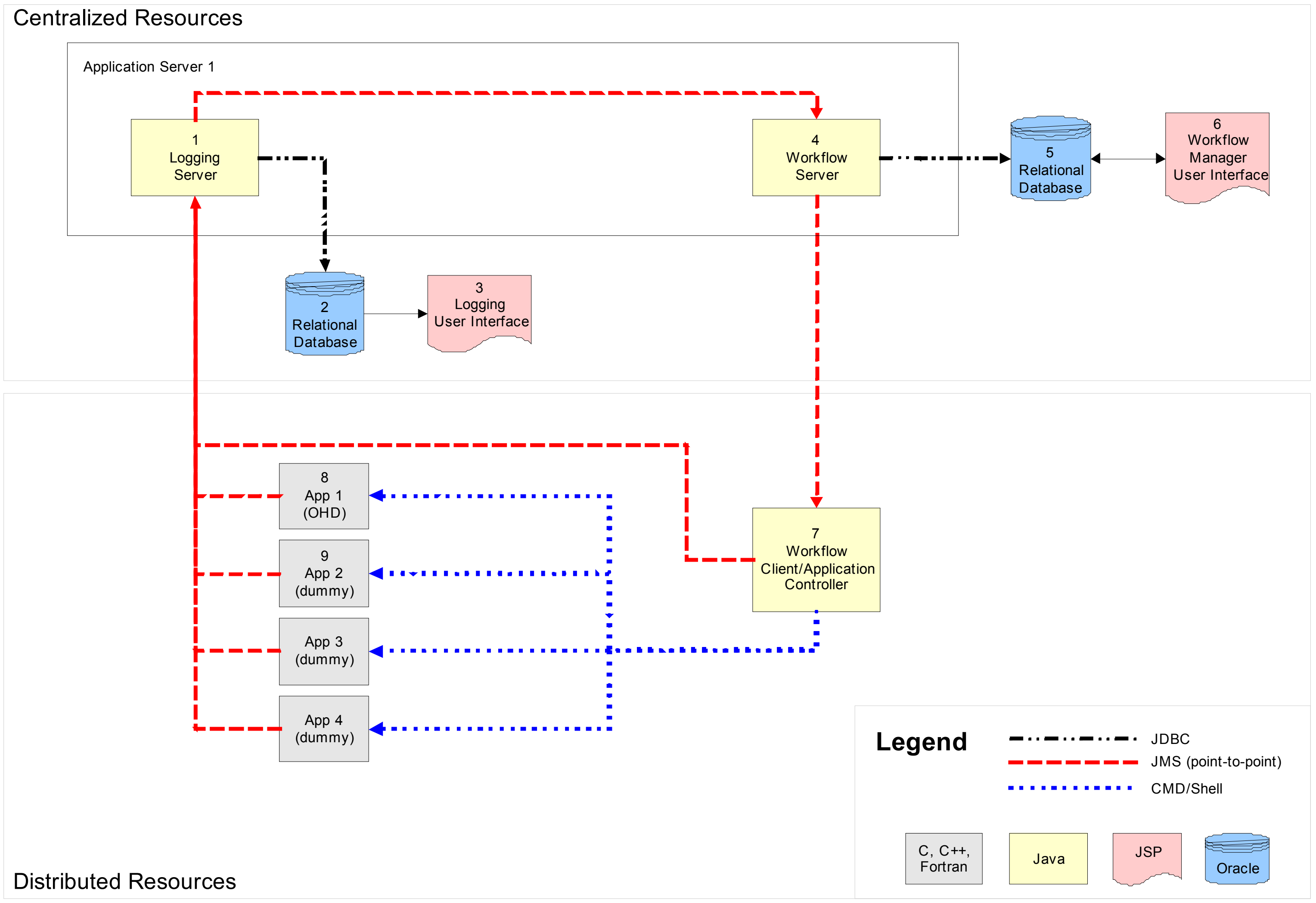
Overview of the System Components

- A workflow server
- A workflow client for each workstation in the Workflow Management System (WMS)
- One or more messaging-enabled applications (“me-apps”)
- A logging server
- A message delivery framework

Overview of Underlying Technology

- J2EE Application Server:
  - Current implementation: BEA WebLogic
  - Alternatives: JBoss (open source), IBM WebSphere, Sun ONE
- Database server:
  - Current implementation: Oracle
  - Alternatives: Informix, PostgreSQL, SQL Server
- Supported Operating Systems:
  - Linux
  - Windows

General Architectural Overview



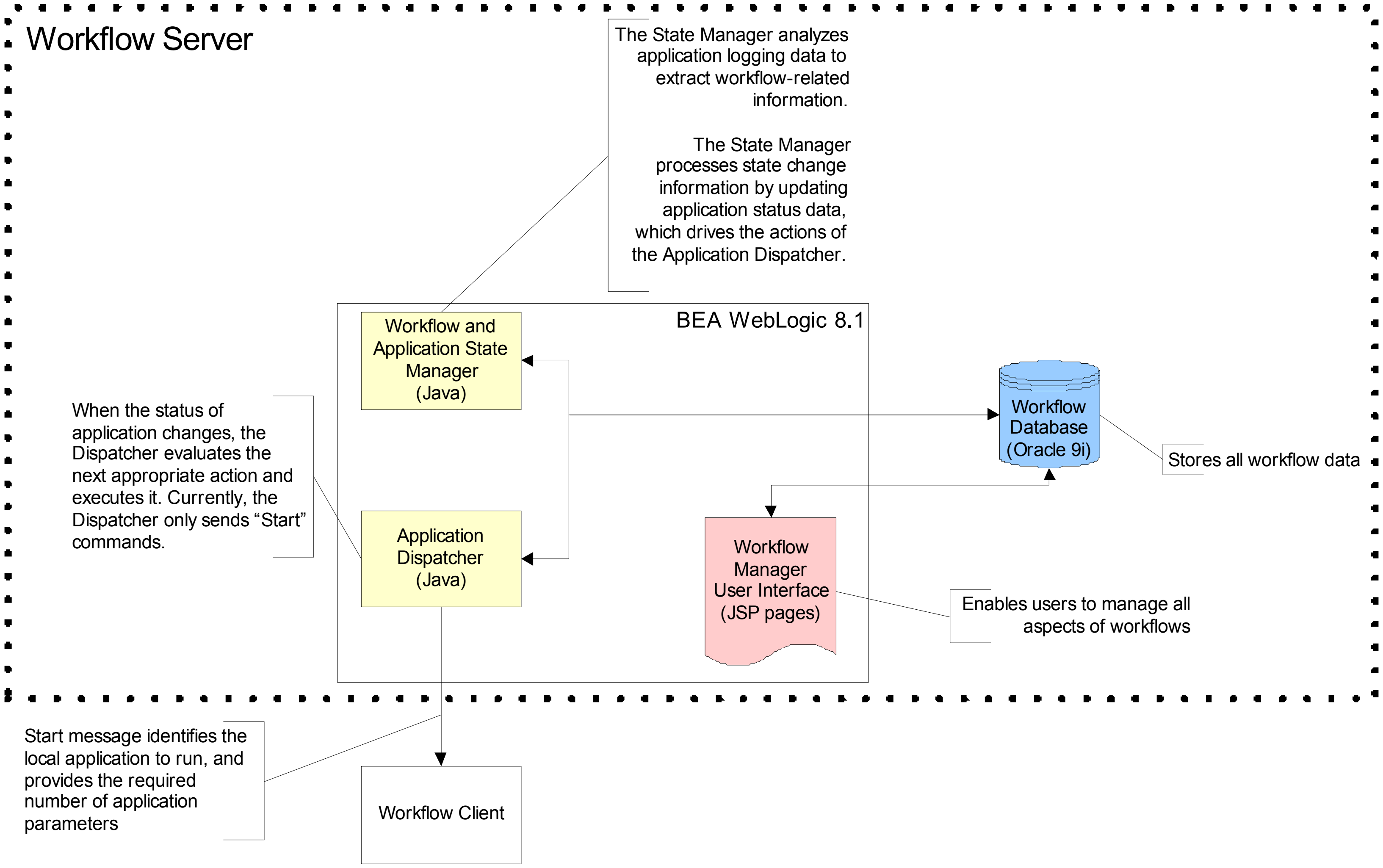
Component Detail: Workflow Server

- Terminology
  - "Workflow": A set of applications that execute according to a defined sequence
  - "Workflow Template": a pre-defined workflow that contains a specific set of applications in a specific order. Templates are saved and re-used when creating workflows
- The workflow server manages
  - Registered workstations
  - Registered messaging-enabled applications
  - Workflow templates
  - Workflows

Workflow Server, continued

- The workflow server supports
  - Creating and editing workflow templates
  - Creating workflows based on existing workflow templates
  - Starting and stopping workflows
  - Viewing workflow status
  - Links into the central application log provided by the Logging Server for specific workflow applications
  - Normal and abnormal application termination
  - Transmission of application control messages to workflow client

Workflow Server, continued


Workflow Server
The State Manager analyzes application logging data to extract workflow-related information.
The State Manager processes state change information by updating application status data, which drives the actions of the Application Dispatcher.
BEA WebLogic 8.1
Workflow and Application State Manager (Java)
Workflow Database (Oracle 9i)
Stores all workflow data
When the status of application changes, the Dispatcher evaluates the next appropriate action and executes it. Currently, the Dispatcher only sends "Start" commands.
Application Dispatcher (Java)
Workflow Manager User Interface (JSP pages)
Enables users to manage all aspects of workflows
Start message identifies the local application to run, and provides the required number of application parameters
Workflow Client

Component Detail: Workflow Client

- Terminology
  - "Workflow Client": A daemon/service application that runs on a registered workstation, receives messages from the Workflow Server and launches messaging-enabled applications
  - "Messenger API": A set of standardized logging functions developed jointly by OHD and Apex. The API has been enhanced to include XML encoding of log messages as well as JMS transmission
  - "Messaging-Enabled Application": An application that uses the messenger API to communicate with the logging server
- The workflow client supports
  - Receiving XML-encoded start messages from the Workflow Server
  - Decoding the start message and comparing start information with local, XML-based application configuration data
  - Starting the indicated application in a child process
  - Running on any number of distributed workstations for concurrent processing

Workflow Client, continued

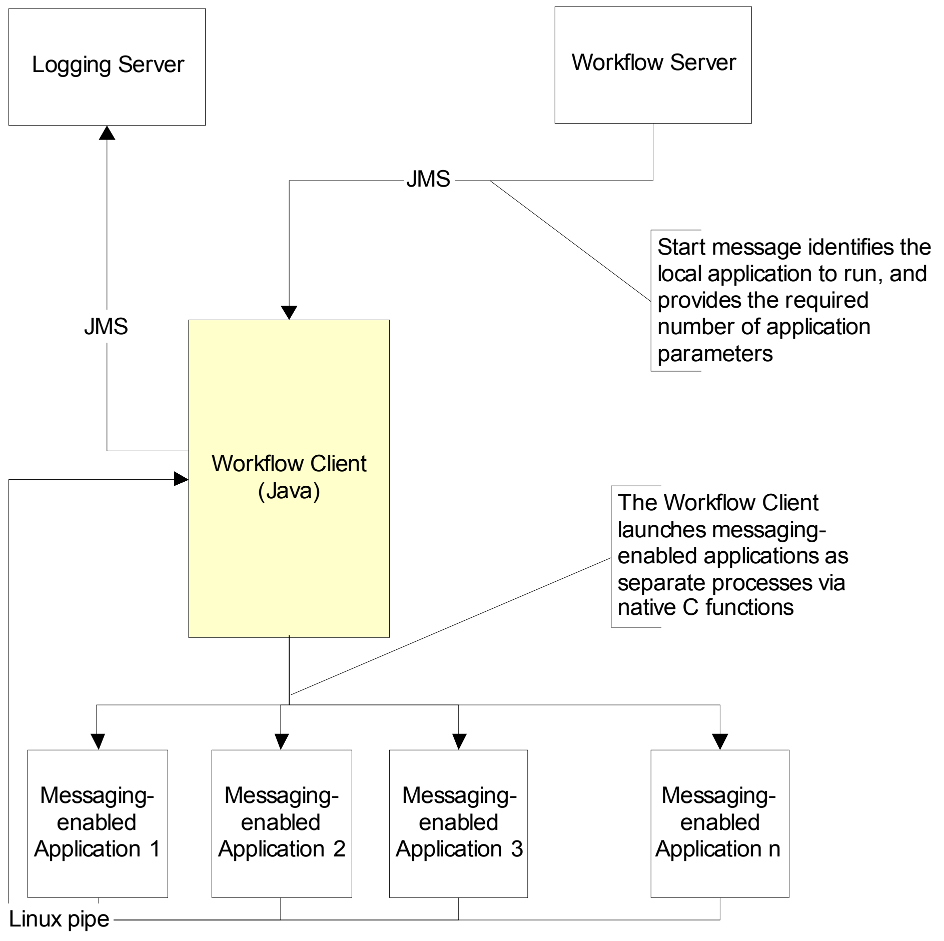
- Sample application start message:

```
<?xml version="1.0" ?>
 <COMMAND APPLICATION-ID="2" LOG-SESSION-ID="559" LOG-SESSION-POSITION="3">
      <ARGUMENT ORDER-ID="0" ARG-TEXT="1"/>
      <ARGUMENT ORDER-ID="1" ARG-TEXT="10"/>
      <ARGUMENT ORDER-ID="2" ARG-TEXT="1"/>
 </COMMAND>
```

- Sample application configuration file:

```
<?xml version="1.0" ?>
   <APPS>
      <APP APPLICATION-ID="1" APPLICATION-NAME="Test_App_01" EXE-
    LOCATION="/fs/shared/home/sliu/DEMO/APPS" LOG-SESSION-POSITION="2"
      />
      <APP APPLICATION-ID="2" APPLICATION-NAME="Test_App_02" EXE-
    LOCATION="/fs/shared/home/sliu/DEMO/APPS" LOG-SESSION-POSITION="3"
      />
      <APP APPLICATION-ID="3" APPLICATION-NAME="Test_App_03" EXE-
    LOCATION="/fs/shared/home/sliu/DEMO/APPS" LOG-SESSION-POSITION="4"
      />
    </APPS>
```

Workflow Client, continued


Logging Server
Workflow Server
JMS
Start message identifies the local application to run, and provides the required number of application parameters
JMS
Workflow Client (Java)
The Workflow Client launches messaging-enabled applications as separate processes via native C functions
Messaging-enabled Application 1
Messaging-enabled Application 2
Messaging-enabled Application 3
Messaging-enabled Application n
Linux pipe

Component Detail: Messaging-enabled Applications

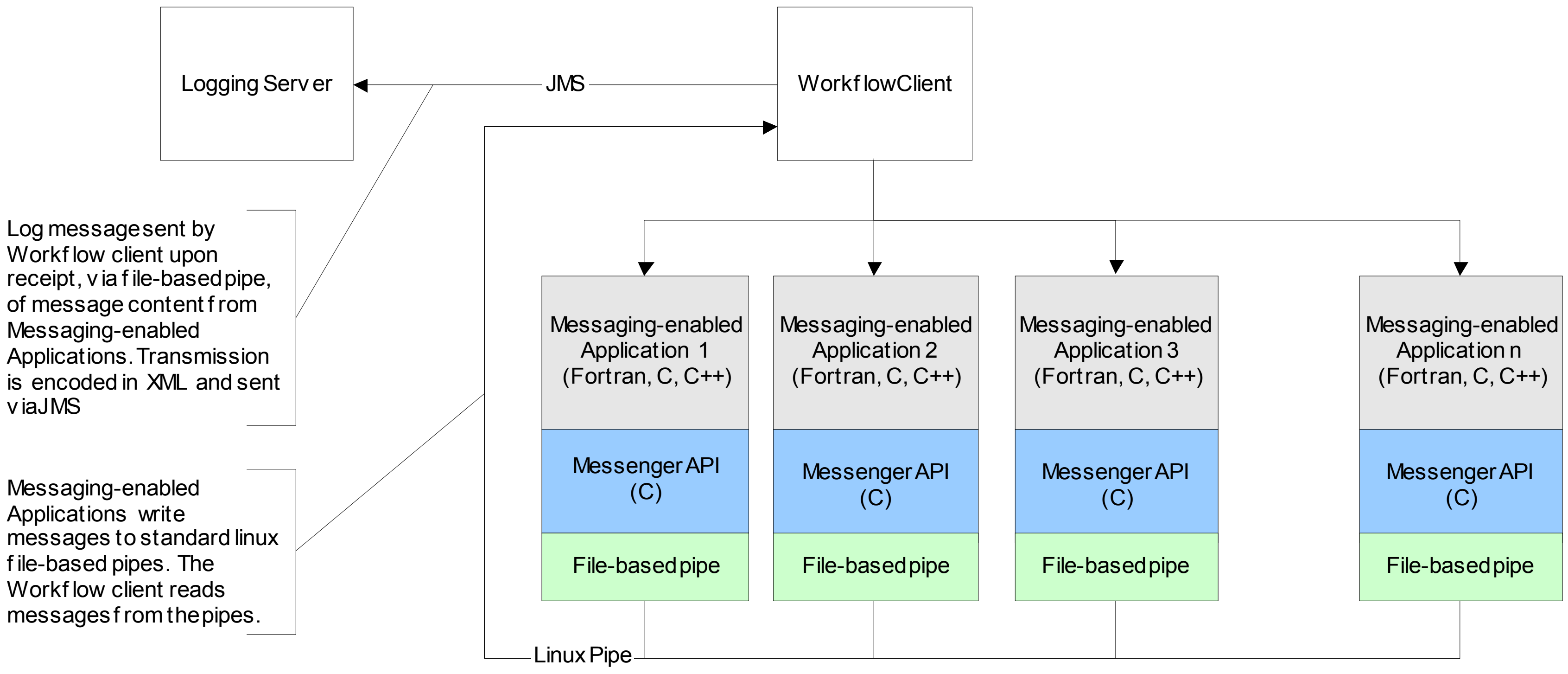
- Terminology
  - "Log Message": messages that follow the messenger API standard, and include additional process management data (see example on following page)
- Messaging-enabled applications support
  - Sending all log messages to the logging server via the messenger API
  - Startup either via C native interface or manually from the command line

Messaging-enabled Applications, continued

- Sample XML log message:

```
<?xml version="1.0"?>
<message session-id="544" process-id="32141">
          <bytecode event="3" severity="1" dataquality="0" problemstatus="0"
trace="10" />
          <timestamp datetime="293864928734" microsecs="142834" />
          <workstation hostname="APEXDEMO" IP="172.16.12.145" />
          <application name="fcst_ens_compu" version="1.0" type="MEAPP" />
          <text>Application fcst_ens_compu finished successfully.</text>
</message>
```

Messaging-enabled Applications, continued


Logging Server
JMS
WorkflowClient
Messaging-enabled Application 1 (Fortran, C, C++)
Messaging-enabled Application 2 (Fortran, C, C++)
Messaging-enabled Application 3 (Fortran, C, C++)
Messaging-enabled Application n (Fortran, C, C++)
Messenger API (C)
File-based pipe
Linux Pipe

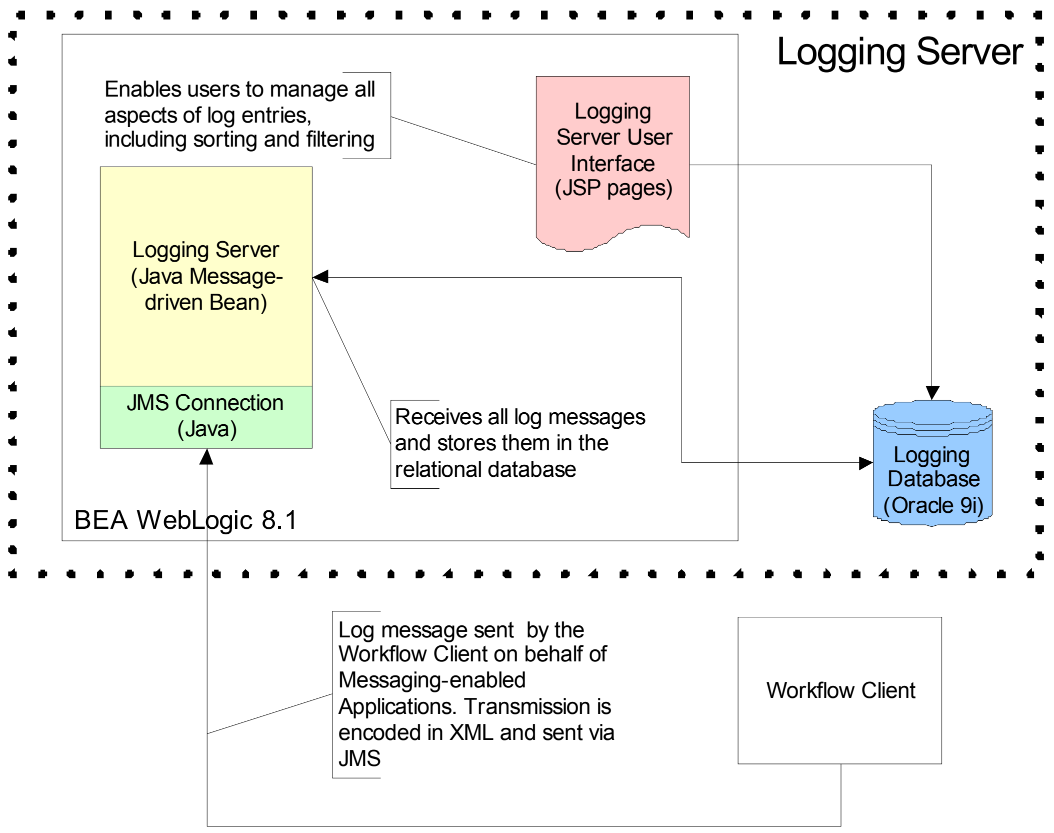
Log message sent by Workflow client upon receipt, via file-based pipe, of message content from Messaging-enabled Applications. Transmission is encoded in XML and sent via JMS

Messaging-enabled Applications write messages to standard linux file-based pipes. The Workflow client reads messages from the pipes.

Component Detail: Logging Server

- Terminology
  - “Log Session”: a continuous session representing log messages from one messaging-enabled application
- The logging server supports
  - Receiving XML-encoded log messages from any messaging-enabled application running with or without workflow management support.
  - Decoding log messages and storing them in a relational database.
  - Displaying log messages, filtered and sorted by any of the data elements.
  - Alerting the workflow server of events that affect workflow management.

Logging Server, continued


Logging Server
Enables users to manage all aspects of log entries, including sorting and filtering
Logging Server User Interface (JSP pages)
Logging Server (Java Message-driven Bean)
JMS Connection (Java)
Receives all log messages and stores them in the relational database
Logging Database (Oracle 9i)
BEA WebLogic 8.1
Log message sent by the Workflow Client on behalf of Messaging-enabled Applications. Transmission is encoded in XML and sent via JMS
Workflow Client

Component Detail: Message-Delivery Framework

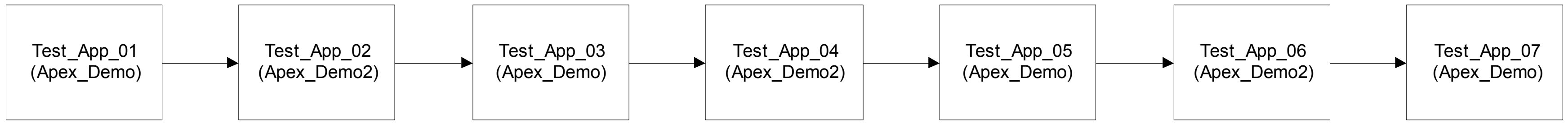
- Terminology
  - “J2EE Application Server”: An application management environment compliant with the Java 2 Enterprise Edition standards. Apex is currently using BEA WebLogic 8.1 as the application server
  - “JMS”: Java Message Service, part of the J2EE specifications
  - “JMS Queue”: A message management object managed by the application server that can receive messages from message producers in a serial, transacted and persisted fashion, as well as deliver the same messages to message consumers
  - “Point-to-Point Queue” (PTP): A JMS queue between two and only two points. One message consumer can write read one message at a time to the PTP queue, and one message consumer can read the same message from the queue. Any message that is read in this manner is deleted from the queue. PTP messages are delivered once and only once

Message-Delivery Framework, continued

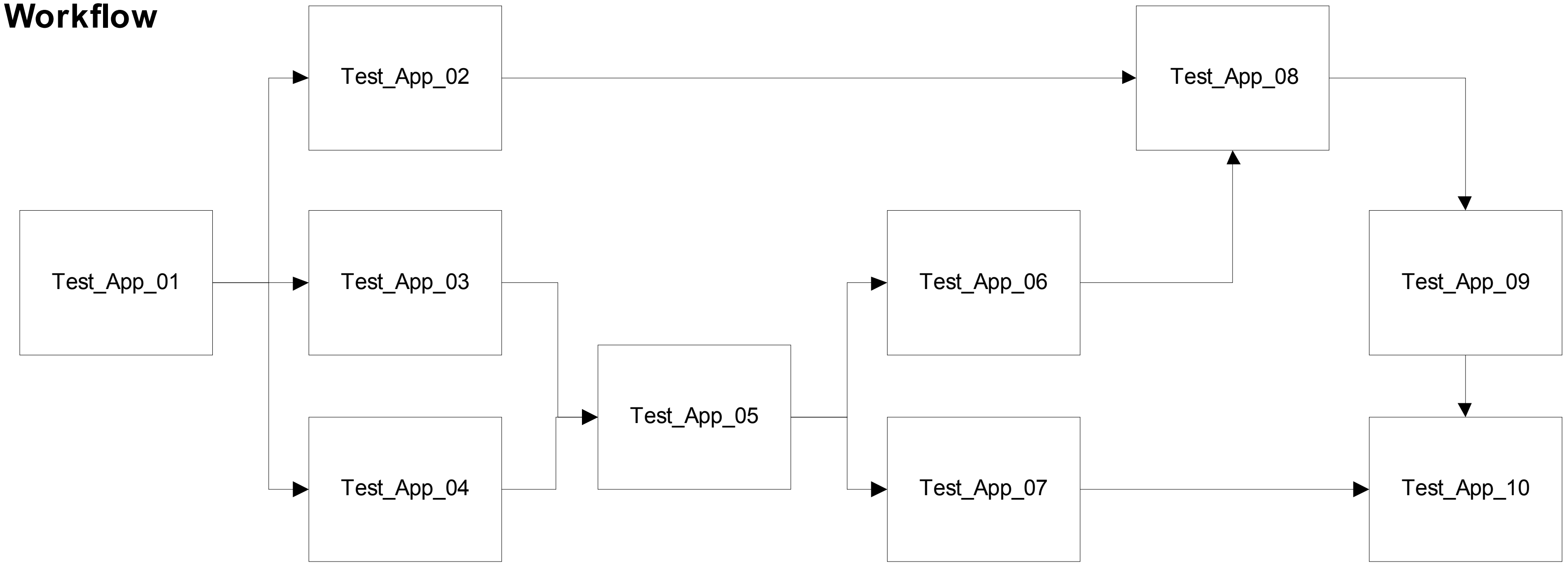
- The Message-Delivery Framework supports
  - Stable, operating system-independent, transacted exchange of messages
  - The capability to exchange messages with servers inside and outside the local network
  - A standard, vendor-neutral API that can run on a variety of commercial and open-source platforms

Example Workflows

Example 1: Simple Sequential Workflow, Running on 2 Different Workstations



Example 2: Sequential and Concurrent Workflow



Conclusions and Next Steps

- Strengths and limitations of the proof-of-concept NWS Workflow Management System
- Strengths and limitations of the underlying architecture
- Next steps

Strengths and Limitations of the Proof-of-Concept WMS

- Strengths
  - Demonstrates functionality of most high-level architectural priorities
  - Built on leading commercial J2EE platform
  - Built according to production-level design concepts
  - Enables multiple deployment scenarios:
    - One centralized logging and workflow server for all RFCs
    - Multiple, distributed logging and workflow servers for some or all RFCs
    - One logging and one workflow server for each RFC
  - Enables organic, gradual upgrade of applications to messaging-enabled protocol, allowing continuous operation

Strengths and Limitations of the Proof-of-Concept WMS

- Limitations
  - As a proof-of-concept system, several areas of the system need further development or rebuilding, specifically
    - Security
    - Error handling and recovery
    - Time-out management by workflow client
    - Integration of workflow management tools with applications that are managed from a user-interface (non-batch applications)
    - Scalability

Strengths and Limitations of the Underlying Architecture

- Strengths
  - Fully distributed, operating system-independent workflow environment
  - Workflows can replace CRON or manual startup of individual applications
  - Flexible workflow configuration
  - Integrated, consolidated logging capability
  - Uses existing applications with only minor modification
  - Integrates relational databases for operation

Strengths and Limitations of the Underlying Architecture

- Limitations
  - More complex operating environment
  - Remote connections can fail
  - Stability and performance of high-volume JMS traffic needs to be established

Next Steps

- Establish run-time environment at NWS OHD to enable testing and test development by HSEB staff
- Upgrade several applications for messaging-enabled operation and test such operation extensively
  - Show that multiple applications can run in the proof-of-concept environment
  - Perform timing studies
- Evaluate the suitability of the proposed architecture for future improvements to NWSRFS

Next Steps, cont.

- Plan, design, and implement improved data management capabilities
  - Implement enhanced data management routines to improve record locking capabilities
  - Transition one or two repositories to management by a relational database management system
- Expand the workflow server to include hydrology-driven workflow model
  - In addition to application workflows, it will be important for the architecture to support dynamically created workflows that are based on hydrologic objects. E.g., this might involve processing forecast points along river flow in the proper sequence
  - Hydrology-driven workflow processing can function as a basis for dynamic, automated forecasting in a distributed modeling environment

Implications of the Proof-of-Concept WMS for the future of NWSRFS

- Enables implementation of a comprehensive application logging system
- Enables implementation of concurrent processing using existing applications
- Improves application performance and data locking challenges due to workflow-driven process control
- Enables RFC developers to test new code against standard workflows without requiring full application recompilation
- Lays groundwork for improved data and lock management via data services or thin-client database connections to relational databases
- Lays groundwork for highly flexible, dynamic distributed modeling capabilities which will likely require concurrent processing on multiple workstations.