BEA WebLogic Server®

Understanding the WebLogic Diagnostic Framework

Version 9.0
Document Revised: July 22, 2005

Copyright

Restricted Rights Legend

Trademarks or Service Marks

Contents

1. Introduction and Roadmap

What Is the WebLogic Diagnostic Framework? 1-1

Document Scope and Audience 1-2

Guide to This Document 1-3

Related Documentation 1-3

2. Overview of the WLDF Architecture

Overview of the Framework 2-1

Data Creation and Collection 2-2

Archival 2-4

Watch and Notification 2-4

Access 2-5

Management 2-6

Diagnostic Image Capture 2-7

How It All Fits Together 2-8

CHAPTER 1

Introduction and Roadmap

This section describes the contents and organization of this guide and the audiences to which it is addressed—*Understanding the WebLogic Diagnostic Framework*.

- "What Is the WebLogic Diagnostic Framework?" on page 1-1
- "Document Scope and Audience" on page 1-2
- "Guide to This Document" on page 1-3
- "Related Documentation" on page 1-3

What Is the WebLogic Diagnostic Framework?

The WebLogic Diagnostic Framework (WLDF) is a monitoring and diagnostic framework that defines and implements a set of services that run within the WebLogic Server® process and participate in the standard server life cycle. Using WLDF, you can create, collect, analyze, archive, and access diagnostic data generated by a running server and the applications deployed within its containers. This data provides insight into the run-time performance of servers and applications and enables you to isolate and diagnose faults when they occur.

WLDF includes several components for collecting and analyzing data, including the following:

- Server Image Capture—creates a diagnostic snapshot from the server that can be used for post-failure analysis
- Archiver—captures and persists all data events, log records, and metrics from server instances and applications

- Instrumentation—adds code to WebLogic servers and the applications running on them to execute diagnostic actions at specified locations in the code
- Harvester—captures metrics from run-time MBeans, including Weblogic Server MBeans and custom MBeans
- Watch and Notification—provides the means for monitoring server and application states and sending notifications based on criteria set in the watches
- Logging services—manages logs for monitoring server, subsystem, and application events. The BEA WebLogic Server® logging services are documented separately from the rest of the WebLogic Diagnostic Framework. See *Configuring Log Files and Filtering Log Messages*.

WLDF provides a set of standardized application programming interfaces (APIs) that enable dynamic access and control of diagnostic data, as well as improved monitoring that provides visibility into the server. ISVs can use these APIs to develop custom monitoring and diagnostic tools for integration with WLDF.

WLDF is a new feature in WebLogic 9.0. In previous releases of WebLogic Server, access to diagnostic data by monitoring agents—which were developed by customers or third-party tools vendors—was limited to JMX attributes, and changes to monitoring agents required server shut down and restart. However, WLDF enables dynamic access to server data through standard interfaces, and the volume of data accessed at any given time can be modified without shutting down and restarting the server.

Document Scope and Audience

This document provides an overview and describes the architecture of the WebLogic Diagnostic Framework (WLDF).

WLDF provides features for monitoring and diagnosing problems in running WebLogic server instances and clusters and in applications deployed to them. Therefore, the information in this document is directed both to system administrators and to application developers. It also contains information for third-party tools developers who want to build tools to support and extend WLDF.

It is assumed that readers are familiar with Web technologies and the operating system and platform where BEA WebLogic Server® is installed.

Guide to This Document

This document is organized as follows:

- This chapter, "Introduction and Roadmap," introduces the organization of this guide and the audiences to which this guide is addressed.
- Chapter 2, "Overview of the WLDF Architecture," describes the WLDF architecture and the functions of and interactions between the components.

Related Documentation

- *Configuring and Using the Weblogic Diagnostic Framework* provides complete documentation for configuring and using the WebLogic Diagnostic Framework. This document should be your primary source for understanding, configuring, and using WLDF.
- *Configuring Log Files and Filtering Log Messages* describes how to use WLDF logging services to monitor server, subsystem, and application events.
- "Configure the WebLogic Diagnostic Framework" in the *Administration Console Online Help* tells how to use the visual tools in the WebLogic Administration Console to configure WLDF.

CHAPTER 2

Overview of the WLDF Architecture

The WebLogic Diagnostic Framework (WLDF) consists of a number of components that work together to collect, archive, and access diagnostic information about the server and the applications it hosts. The framework and the components are described in the following sections:

- "Overview of the Framework" on page 2-1
- "Data Creation and Collection" on page 2-2
- "Archival" on page 2-4
- "Watch and Notification" on page 2-4
- "Access" on page 2-5
- "Management" on page 2-6
- "Diagnostic Image Capture" on page 2-7
- "How It All Fits Together" on page 2-8

Overview of the Framework

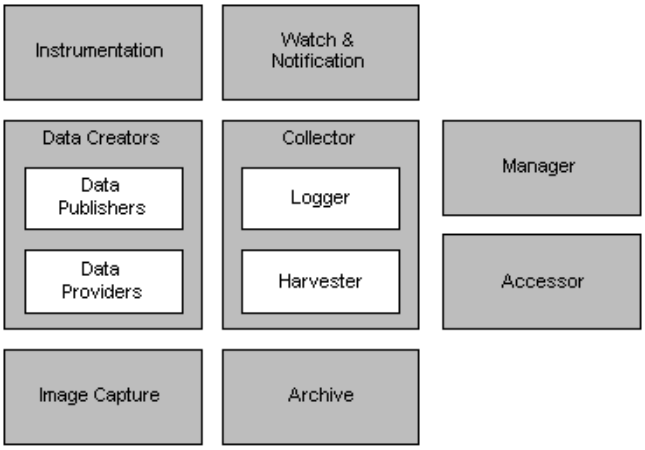
The WLDF architecture comprises the following components:

- Data creators (data publishers and data providers)
- Collectors (Logger and Harvester)
- Archive

- Data Accessor
- Watch and Notification
- Manager
- Image Capture

Data creators generate diagnostic data that is consumed by the Collector. The Collector coordinates with the Archive to persist this data and with the Watch and Notification system to provide automated monitoring. The Accessor interacts with the Collector to expose current diagnostic data and with the Archive to present historical data. The Manager provides a configuration and control interface for managing the framework. Finally the Image Capture facility provides a model for capturing a diagnostic snapshot of key server state. The relationship among these components is shown in Figure 2-1.

Figure 2-1 Major WLDF Components



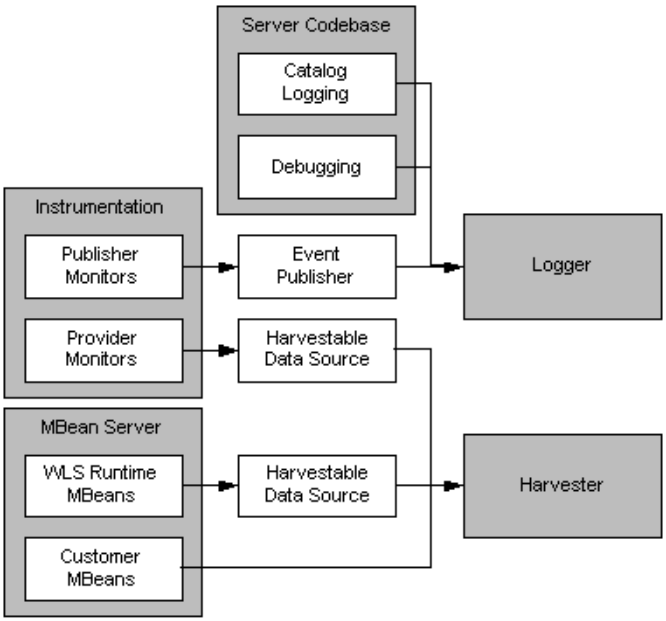
All of the framework components operate at the server level and are only aware of server scope. All the components except for the Manager exist entirely within the server process and participate in the standard server lifecycle. All artifacts of the framework are configured and stored on a per server basis.

Data Creation and Collection

Diagnostic data is collected from a number of sources. These sources can be logically classified as either *data providers*, data creators that are sampled at regular intervals to harvest current

values, or *data publishers*, data creators that synchronously generate events. Data providers and data publishers are distributed across components, and the generated data can be collected by the Logger and/or by the Harvester, as show in Figure 2-2, and explained below.

Figure 2-2 Relationship of Data Creation Components to Data Collection Components



Invocations of the server logging infrastructure serve as inline data publishers, and the generated data is collected as events. (The logging infrastructure can be invoked through the catalog infrastructure, the debugging model, or directly through the Logger.)

The Instrumentation system creates monitors and advice—some of which are publishers and some of which are providers—and inserts them at well-defined points in the flow of execution. Publishers generate events that are consumed by the traditional logging framework. Providers expose their data to the Harvester using the data source interface.

Components registered with the MBean Server may also make themselves know as data providers by registering with the Harvester. All providers registered with the Harvester are then eligible for collection based on the current harvesting configuration, which is dynamically

controllable through the management interface. Collected data is then exposed to both the Watch and Notification system for automated monitoring and to the Archive for persistence.

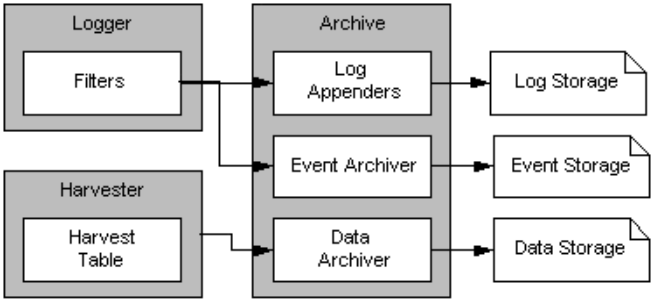
Archival

Past state is often critical in diagnosing faults in a system. This requires that state be captured and archived for future access, creating a historical archive. In WLDF the Archive meets this need with several persistence components. Both events and harvested metrics can be persisted and made available for historical review.

Traditional logging information, which is human readable and intended for inclusion in the server log, is persisted through the standard logging appenders. New event data that is intended for system consumption is persisted into an event store using an event archiver. Metric data is persisted into a data store using a data archiver. The relationship of the Archive to the Logger and the Harvester is shown in Figure 2-3.

The Archive provides access interfaces so that the Accessor may expose any of the persisted historical data.

Figure 2-3 Relationship of the Archive to the Logger and the Harvester

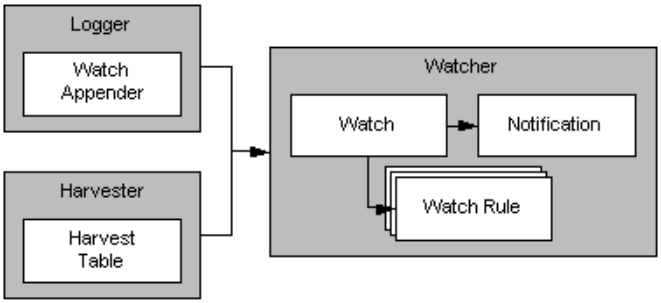


Watch and Notification

The Watch and Notification system can be used to create automated monitors that observe specific diagnostic state and send notifications based on configured rules.

A watch rule can monitor either event data from a data publisher or metric data from a data provider that is harvested by the Harvester. The Watcher is capable of managing watches that are composed of a number of watch rules. These relationships are shown in Figure 2-4.

Figure 2-4 Relationship of the Logger and the Harvester to the Watch and Notification System



One or more notifications can be configured for use by a watch. By default, every watch logs an event in the server log. In addition SMTP, SNMP, JMX, and JMS notifications are supported.

Access

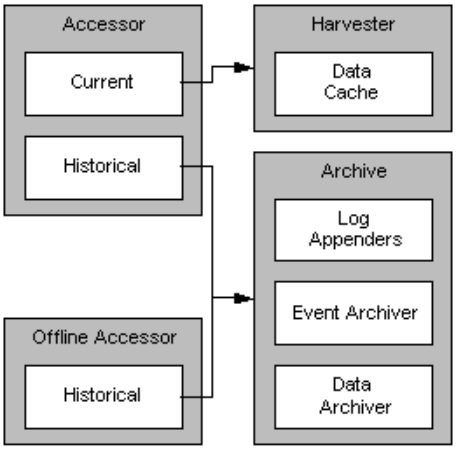
The Accessor provides access to all the data collected by WLDF, including event and metric data. The Accessor interacts with the Harvester to get the current state of harvestable values in the server. The Accessor interacts with the Archive to get historical sate including logged event data and persisted metrics.

When accessing data in a running server, a JMX-based access service is used. The Accessor provides for data lookup by type, by component, and by attribute. It permits time-based filtering and in the case of events filtering by severity, source and content.

Tools may wish to access data that was persisted by a server which is not currently active. In these cases an offline Accessor is also provided. It supports access to historical data only, as no current state exists.

The relationship of the Accessor to the Harvester and the Archive is shown in Figure 2-5.

Figure 2-5 Relationship of the Online and Offline Accessors to the Harvester and the Archive



Management

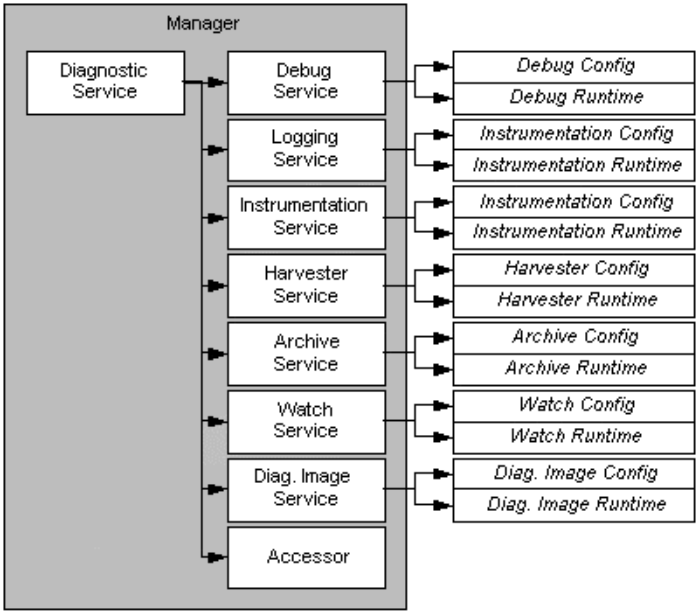
The Manager provides the management interface of the entire framework. It provides access to a server-specific top-level Diagnostic Service which then provides references to the appropriate component-specific services. Each service then references the appropriate configuration component and the appropriate runtime component running in the content of the server of interest.

The management interface is all JMX-based and relies heavily on service-oriented interfaces. All of the service components are defined as runtime Mbeans. The service components exist in the administration server and interact with runtime components that exist in the server process associated with the framework instance.

In addition to management services, the Diagnostic Service provides diagnostic data about the framework itself and a reference to the Accessor for the given server.

The relationship of the Manager's diagnostic services to the configuration and runtime components is illustrated in Figure 2-6.

Figure 2-6 Relationship of the Manager's Services to Configuration and Runtime Components

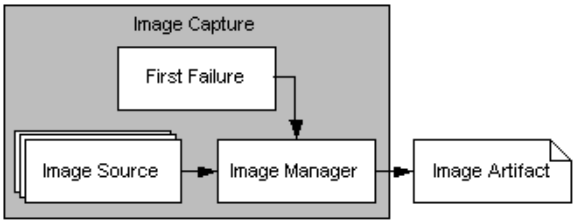


Diagnostic Image Capture

The Diagnostic Image support gathers the most common sources of key server state used in diagnosing problems and packages that state into a single artifact that can be made available to support, as shown in Figure 2-7. The diagnostic image is, in essence a diagnostic snapshot or dump from the server.

The image capture support includes both an on-demand capture process and an automated capture based on some basic failure detection.

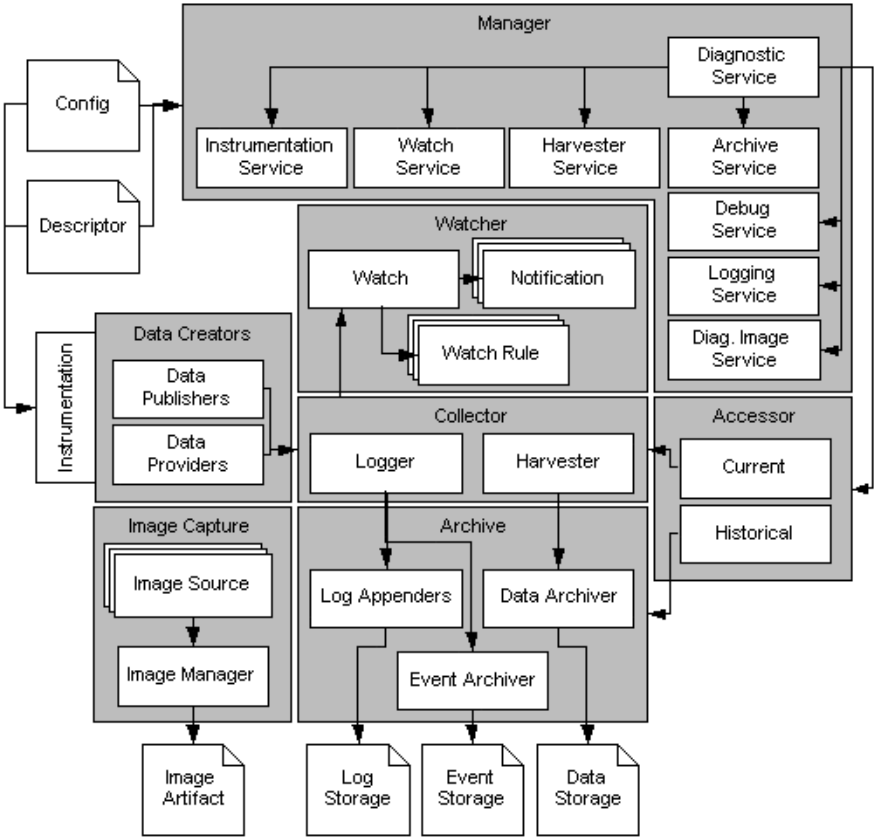
Figure 2-7 Diagnostic Image Capture



How It All Fits Together

Figure 2-8 shows how all the parts of WLDF fit together.

Figure 2-8 Overall View of the WebLogic Diagnostic Framework



A
Accessor component, about 5
Archiver component, about 1, 4
C
components, relationship among 8
configuring logging services 3
configuring WLDF 3
D
data collection 2
data creation 2
Diagnostic Image Capture component, about 7
document audiences 2
dynamic access
    to server data 2
F
framework, overview of WLDF 1
H
Harvester component, about 2
I
Instrumentation component, about 2
L
Logging services, about 2
M
Manager component, about 6
S
Server Image Capture component, about 1
W
Watch and Notification component, about 2, 4
WebLogic Diagnostic Framework. See WLDF
WLDF, about 1