The Applied Informatics Open Service Platform

Overview

Version 1.3

Purpose of This Document

This document gives an overview of the Applied Informatics Open Service Platform,

The document is targeted at developers and development/technical managers wanting to get an overview of the functionality and features offered by the Applied Informatics Open Service Platform. Familiarity with the C++ programming language and with the POCO C++ Libraries is assumed.

Validity of This Document

This document covers release 1.0 and later releases of the Applied Informatics Open Service Platform.

Copyright, Trademarks, Disclaimer

Table of Contents

1 Executive Summary

This document gives an overview of the Applied Informatics Open Services Platform. The Open Service Platform (OSP) is a C++ based middleware providing a plug-in and services-based environment for developing, deploying, running and managing modular network-based applications. As such, OSP is to C++ to what the OSGi™ Service Platform[1] is to Java.

OSP is based on the POCO C++ Libraries[2]. The POCO C++ Libraries are a collection of open-source class libraries that simplify the development of network-centric, portable applications in C++. The libraries integrate perfectly with the C++ Standard Library and fill many of the functional gaps left open by it. Using the POCO C++ Libraries as a foundation makes OSP available on a variety of platforms, from Windows and Solaris based enterprise systems to Linux and QNX-based embedded systems.

At the core of OSP lies a powerful software plug-in model based on the concept of bundles. A bundle is a deployable entity, consisting of both executable code and the necessary configuration, data and resource files required for running the code. Bundles extend the functionality of an application by providing features to other bundles, end-user functionality or web services. A central Service Registry allows bundles to discover the services provided by other bundles.

Bundles can be added, updated, started, stopped or removed from an application without the need to terminate and restart the application.

This plug-in based architecture of OSP addresses an increasing problem in software development: The large number of application configurations that need to be developed and maintained. The standardized OSP component architecture simplifies this configuration process significantly.

[1] OSGi is a trademark or a registered trademark of the OSGi Alliance in the United States, other countries, or both. See http://www.osgi.org for more information on the OSGi Service Platform.
[2] See http://pocoproject.org for more information on the POCO C++ Libraries

2 Introduction

Today's applications are becoming ever more complex. A large part of this complexity stems from the need to support different configurations, e.g. for different devices, operating system environments or customer requirements. Huge monolithic applications like the ones developed in the past do not (or only at an enormous cost) provide the necessary flexibility required today.

Many companies have noticed this, and are basing their latest software architectures on a dynamic plug-in model. In this model, they implement a very basic skeleton application merely acting as a loader and execution environment (or container) for plug-ins – shared libraries loaded dynamically as required at run-time. All the features of an application are then provided by plug-ins, which can be added to an application on demand. While Java developers have been able – for a few years now – to choose between the various readily available implementations of such a framework based on the OSGi Service Platform specification[3] developed by the OSGi Foundation, companies relying on C++ have no such standard environment readily available and are forced to implement their own system.

This is where the Applied Informatics Open Service Platform (OSP) comes in. OSP is a C++ based middleware providing a service-oriented and plug-in based environment for developing, deploying, running and managing modular network-based applications. As such, OSP is to C++ to what the OSGi Service Platform is to Java. In fact, the design of OSP draws many ideas and inspirations from the OSGi Service Platform.

OSP is based on the POCO C++ Libraries[4]. The POCO C++ Libraries are open-source class libraries that simplify the development of network-centric, portable applications in C++. The libraries integrate perfectly with the C++ Standard Library and fill many of the functional gaps left open by it.

The classes provided by the POCO C++ Libraries provide support for multi-threading, streams, accessing the file system, shared libraries and class loading, configuration file and command line handling, security, network programming (TCP/IP sockets, HTTP, FTP, SMTP, SSL/TLS, etc.), XML parsing (SAX2 and DOM) and generation, as well as access to SQL databases. Applied Informatics provides various additional C++ class libraries based on the POCO C++ Libraries, providing features such as distributed objects and web services, automatic discovery of network services, as well as remote configuration capabilities based on the NETCONF protocols.

Using the POCO C++ Libraries as a foundation makes OSP available on a variety of platforms, from Windows and Solaris based enterprise systems to embedded Linux or QNX-based embedded systems. Applications based on the POCO C++ Libraries – and therefore, applications based on OSP – can

[3] http://www.osgi.org
[4] See http://pocoproject.org for more information on the POCO C++ Libraries

be compiled for and executed on different platforms, all from the same source code base.

This component-based architecture of OSP addresses an increasing problem in software development: The large number of application configurations that need to be developed and maintained. The standardized OSP component architecture simplifies this configuration process significantly.

3 OSP Architecture

The Open Service Platform is based on a layered architecture, depicted in Figure 1. At the core of OSP is the Portable Runtime Environment, consisting of the C and C++ standard libraries and the POCO Core Libraries (Foundation, XML, Util and Net). Layered above the Portable Runtime Environment is the OSP Framework, consisting of Service Registry, Life Cycle Management, Bundle Management and Standard Services. Application-specific bundles based on the OSP Framework implement the actual application logic.

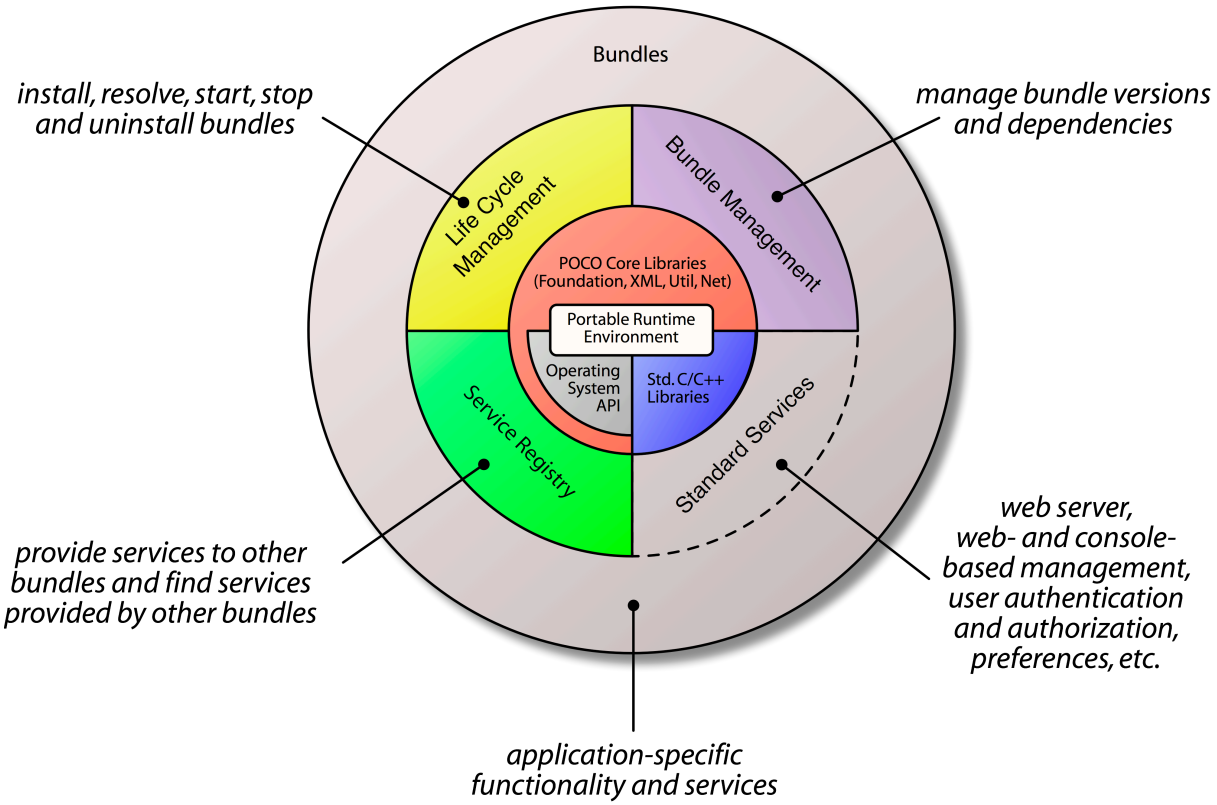

Figure 1: Open Service Platform layered architecture

3.1 Portable Runtime Environment

The Portable Runtime Environment sits at the center of the OSP architecture. Based on the C and C++ Standard Libraries, as well as on the POCO Core Libraries, it provides platform-independent low-level services to the upper OSP layers, such as:

- access to the file system
- multithreading support
- shared library and class loading
- notifications and events
- logging
- XML parsing
- configuration data handling
- TCP/IP sockets and support for various network protocols (HTTP, FTP, SMTP, POP3, etc.)
- various utility classes and functions

By isolating applications from the operating system interfaces, the Portable Runtime Environment makes it possible to write applications that can be compiled for and run on different operating system platforms and processor architectures, all from the same source code.

3.2 Service Registry

The Service Registry allows a bundle to register its services to make them available to other bundles, as well as to discover the services provided by other bundles. Since bundles can appear and disappear in an application at any time, the Service Registry also provides notification mechanisms so that a bundle can be informed when another bundles it uses disappears from the system.

A bundle can discover a certain service by using a simple query language to query the Service Registry for a service with certain properties.

3.3 Life Cycle Management

Bundles adhere to a well-defined life cycle, shown in Figure 2. The Life Cycle Management in OSP ensures that every bundle in the system follows this life cycle, which is outlined in the following.

A bundle is installed into an application and starts its life cycle in the Installed state. From the Installed state, a bundle can either be uninstalled, or resolved. Resolving a bundle means determining all its dependencies on other bundles, and verifying that all required bundles are available. Only a successfully resolved bundle can be started. Starting a bundle causes a special class provided by the bundle, the *Bundle Activator*, to be loaded and invoked. The Bundle Activator then takes care of registering the bundle's services and does all the necessary steps so that the bundle can provide its services. After the Bundle Activator has successfully done its work, the bundle is in Active state. Eventually, an active bundle will be stopped again, and possibly removed from the system (uninstall).

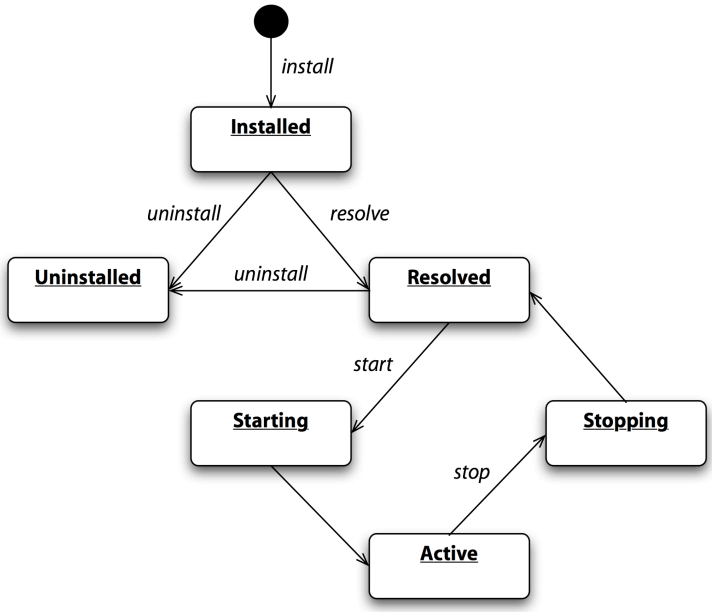

Figure 2: The bundle life cycle

3.4 Bundle Management

The Bundle Management layer in OSP takes care of dealing with the physical storage of bundles in the file system, as well as the loading of shared libraries contained in bundles.

Bundles are a collection of files (manifest, shared libraries, data files, configuration files, resource files, etc.) stored in a certain directory structure. For easier deployment, a bundle can be packed into an archive, using the Zip file format. In a future version of OSP it will also be possible to cryptographically sign a bundle and to encrypt the contents of a bundle.

Bundles can contain multiple versions of a shared library for different operating system platforms and hardware architectures. Bundle Management ensures that the correct version of a shared library for the current operating system and hardware architecture is loaded. This makes it possible to provide a single bundle file that can be, for example, deployed on both a Windows and a Linux system.

3.5 Standard Services

OSP comes with a number of standard services implementing commonly required features. Examples include a HTTP server, support for sending email messages, Web-based bundle management, a Shell service for building extensible command-line interfaces as well as a configuration and preferences storage service.

4 Portability and Platforms

The Applied Informatics Open Service Platform is available for a variety of platforms.

4.1 Supported Platforms

All major desktop and server platforms are readily supported, including Windows XP/Vista/7, Mac OS X, Linux, HP-UX, Tru64 and Solaris, in addition to embedded Linux, QNX Neutrino and other POSIX-compliant embedded operating systems.

4.2 Porting

Porting the OSP to a new platform can be done by a customer, with optional support from Applied Informatics' developer team. Alternatively, Applied Informatics can be contracted to do the complete porting to a platform of your choice.

4.2.1 Target Platform Requirements

Target platforms for the OSP must meet the following minimum requirements:

- a 32-bit CPU
- a minimum of 16 MB of RAM available for application code
- optional FPU (or software-based floating point support)
- a TCP/IP stack
- multithreading support for the threads and synchronization classes
- filesystem support
- an ANSI/ISO C++ conforming C++ compiler; see next section

4.2.2 C++ Compiler Requirements

To successfully build the Open Service Platform, a C++ compiler must meet the following requirements:

- full ISO/IEC 14882 standards compliance (including templates, exception handling, runtime type identification)
- a conforming implementation of the C++ standard library, including the STL (minimum: strings, I/O streams and containers/iterators)

Following is an incomplete list of compilers (and platforms) known to work with OSP:

- Microsoft Visual C++ 13.10 and 14.0 (Windows 2000/XP/Vista/7/CE)
- GCC 3.3-20030314 (Mac OS X 10.3)
- GCC 4.0 (Mac OS X 10.4, 10.5)
- GCC 4.2 (Mac OS X 10.6)
- GCC 3.3.1 (Linux 2.4.21, QNX Neutrino 6.3)
- GCC 3.4.1 (Linux 2.4.21 and 2.6.3)
- GCC 4.3 (Linux 2.6.x)
- Sun ONE Studio 8 C++ (Solaris 9)
- Compaq C++ 6.5-040 (HP Tru64 5.1b)
- HP ANSI C++ A.03.57 (HP-UX 11.11 PA-RISC)
- HP ANSI C++ A.06.00 (HP-UX 11.23 IA64)

5 Support, Licensing and Contact

For a complete description of the license regulations and support options available for OSP, please visit http://www.appinf.com.

Applied Informatics can be reached at one of the following addresses:

Applied Informatics Software Engineering GmbH

St. Peter 33
9184 St. Jakob im Rosental
Austria

Phone: +43 4253 32596
Fax: +43 4253 32096
E-Mail: info@appinf.com
Web: www.appinf.com