

Professional Edition 6.2

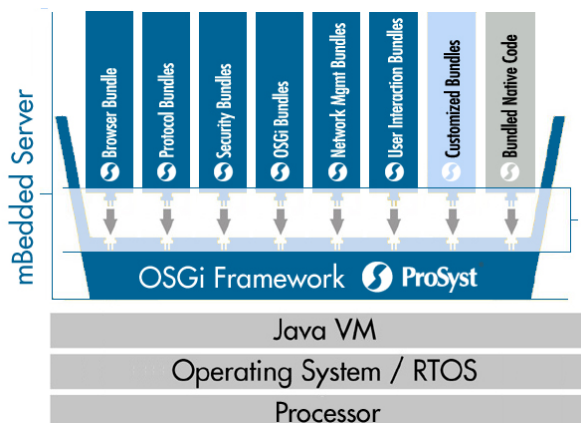
The Professional Edition of mBedded Server (mBS) provides an implementation of the OSGi R4.1 Core Specification and parts of OSGi R4.1 Compendium Specification specifically designed for embedded use. The OSGi technology defines standardized primitives to construct applications from small, reusable and collaborative components. It can be used as base software for building a variety of systems: home gateways, automotive telematics systems, fleetboard systems, and mobile clients like PDAs or smart-phones.

Features

- Compatibility with OSGi Service Platform Core and Compendium Specifications Release 4.1
- Framework enhancements including module architecture, lazy initialization of bundles, resource management, JVM-specific optimizations
- Additional features such as GUI-based administration, control units, data mapper, JSP support
- Key services and extensions for running Eclipse plug-ins and eRCP applications

Benefits

- Modular and flexible architecture ensuring dynamic configuration and update of individual software modules
- High-performance protected runtime environment
- Hardware platform independence
- Small memory footprint and minimal system requirements
- User-friendly GUI administration



mBedded Server Architecture

Compliance with OSGi R4.1 Core and Service Compendium Specifications

The mBedded Server Professional Edition fully implements the OSGi R4.1 Core Specification and supports part of the OSGi R4.1 Service Compendium Specification as well. The OSGi Mobile Specification and the rest of the OSGi Service Compendium Specification are implemented in certain packages of the mBS Extensions.

mBedded Server

- The complete implementations of the Security Layer, Module Layer, Lifecycle Layer, Service Layer and Framework API are forming the core of the mBS Professional Edition Framework. The Framework also supports optional features such as bootclasspath extensions.
- The edition also contains implementation of system services such as Start Level Service, URL Handlers Service, Permission Admin Service, Conditional Permission Admin, and Package Admin Service. Some of the OSGi-defined system services are extended with additional functionality as well.
- The standard OSGi Log Service is enriched with means for persistent storing of log entries, as well as log filtering.
- The Configuration Admin Service is enhanced with initial loading of bundle configurations and utilities for modules wishing to modify their configurations.
- The User Admin Service provides mechanisms for authentication and authorization of users. Its ProSyst extension provides password-based authentication of users and invalidates already authenticated users at logout.
- The IO Connector Service offers a flexible communication mechanism with extensible support for various connection types based on TCP sockets, UDP sockets or HTTP.
- The HTTP Service implementation includes the required HTTP Server, servlet engine and service for registering servlets and resources. ProSyst adds a secure HTTP server and a service to review registered aliases. The JSP runtime support enables the deployment of Java Server Pages as pre-compiled servlets in the HTTP server. The raw JSP files are transferred into Java class files with the help of a special translation tool.

Technical Data Sheet

- The Metatype Service is extended with support for initial loading of bundle configuration data.
- The Declarative Services support is enhanced with the option to store (cache) parsed component descriptions in a database.
- Device Access, Service Tracker, Execution Environment, Preferences Service, XML Parser Service, Wire Admin Service, Measurement and Positioning and Event Admin are implemented as defined by the OSGi Specification. In addition, ProSyst created extensions bringing extra functionality on top of OSGi-defined interfaces.

Framework Enhancements

In order to meet various deployment requirements, ProSyst introduced multiple enhancements to the OSGi Service Platform Specification:

Module Architecture

The ProSyst framework implementation is separated in modules with clear interfaces to easily change the framework behavior. For example, the module for class loading or for security can be configured to satisfy best the needs of the target platform.

Bootstrapping Mechanism

Developers can benefit from the internal bootstrapping mechanism developers by creating customized and optimized initial configurations. In addition, they can integrate own mechanism for bundle installation or provide custom initial bootstrap data.

Storage Abstraction

The storage module abstracts operations with certain files in the framework – system files of the framework, files storing bundle-specific content like JAR files, native libraries, etc. The framework is able

to work with custom implementations of the storage module – for example, an optimized storage that does not store JAR files separately but keeps their content in a database.

Flash Memory Support

The framework provides support for transferring files from flash memory into temporary RAM files and for synchronizing the RAM files with the persistent storage in a background process. This solves the common problem of most embedded devices, which use persistent flash file systems with slow write access.

JVM Specific Optimizations

The framework can manage bundles packed in optimized archive formats, such as J9 JXE files and Tao native images, for speeding up the system startup time and reducing the memory consumption.

Resource Management

Resource management enables monitoring and managing of allocated resources for each bundle running in the framework including active threads and data spaces for any JVM. In addition, when using the mBS Professional Edition with the J9 JVM, it is possible to control bundle memory spaces and sockets as well.

Lazy Initialization of Bundles

Lazy initialization improves the boot time of the framework and of the bundles it runs. The benefit from lazy initialization is that loading of bundle classes is postponed until they are requested.

Tracer and Measurements

For tracking the internal behavior of the framework and core bundles the framework offers the Tracer and Measurement utilities. They provide useful and detailed information of traced or measured activities, such as installation/de-installation of bundles, service registration, event generation, etc.

Data and Error Protection

Special protection of files created by the framework is provided. In this way, data loss on unexpected device restarts is minimized. Error protection prevents from user errors that might leave the system in a non-functional state.

Other Extensions

The framework owns additional functionality important for real commercial systems – certificate storage and trust verification, an abstraction of security policy, support for restarting the framework and OS, critical errors handling, access to core framework operations and integration with mBSA 3.0. The last feature makes possible control and monitoring of the framework by an external agent such as mBSA.

Additional Features Provided as Bundles

ProSyst has developed some valuable bundles on top of the framework:

ProSyst Utilities

A set of powerful utilities such as periodical timer, thread pool, native timer, native process utility, log utility, jar utility, system info utility, string and text formatting utility ease the development and integration of applications on top of the mBS Professional Edition.

DB

ProSyst provides a simple database format for easy storage of tree-like data. Data is structured as nodes. Each node has name and associated value, and can have sub-nodes. The DB manager has special support for flash storages so number of writes can be controlled. To accomplish data isolation, multiple DBs can be created. The DB has also special support for restore points.

PMP Communication

ProSyst Message Protocol (PMP) is a simple and easy to use object RPC protocol that relies on the OSGi IO Connector Service. PMP provides an efficient technique for remote method invocation of services in an OSGi framework from a remote location (for example, from another framework).

It uses no stubs or skeletons, leading to lightweight performance. Communication is independent from the underlying transport and allows using various transport media.

Control Units

ProSyst has developed a common abstraction, which introduces a unified interface to manage different types of controllable modules - these can be devices (physical or virtual) handled by the appropriate drivers, software applications, etc.

All devices attached to the system can be represented as Control Units, and thus can be controlled via a unified interface regardless of their type such as UPnP, LonWorks, KNX, EHS, X10, etc. The generic representation offered by the Control Unit model facilitates the automation and execution of operations on various types of controllable modules.

Abstract Driver

The Abstract Driver is a utility API, which provides a common implementation of a device driver and can be extended by drivers designed for specific protocols. It contains generic logic for registering device control units into the OSGi framework.

A device driver built on top of the Abstract Driver performs the following functions:

- Provides an implementation of a device model and device controller.
- Creates and registers a control unit according to the features of the attached device.

- Presents a common way for extending drivers to declare actions and state variables for managed devices.
- Defines a Device State Graph for the managed device type and watches for possible inconsistencies in the device model.

Data Mapper

Data Mapper is a tool for converting byte arrays to objects and vice versa. The conversion is based on mapping rules described in XML format.

Data Mapper provides a universal way for Java applications to access data from different types of networks and communication protocols. For example, you can use Data Mapper to create XML-based device drivers without the need to write any Java code.

Data Mapper consists of a Java math evaluator, basic Data Mapper and Control Unit extension. The Java math evaluator is used to evaluate boolean conditions and math expressions. The basic Data Mapper includes an XML parser, data mapping elements and an expression evaluator. The parser is used to parse the XML file with mappings. The data mapping elements are used to keep parsed mappings for later evaluation. The expression evaluator is used to evaluate conditions and math expressions. It uses the Java math evaluator as well as a custom functions evaluator for inserting own functions in the expressions.

Equinox Features

Special features defined by Eclipse are included in the Professional Edition of mBedded Server. The aim of supporting these features is to provide the runtime for Eclipse plug-ins and eRCP applications. Another aim is to enable easy porting of user defined/proprietary/open source bundles and applications from Equinox to ProSyst mBedded

mBedded Server

Technical Data Sheet

Server Professional Edition. Equinox features include support for the Equinox framework extension services, common utility classes, extension points registry, and preferences.

XML Parsers

The Professional Edition of mBedded Server contains two popular XML parsers packaged as bundles - Crimson and Xerces.

Tools

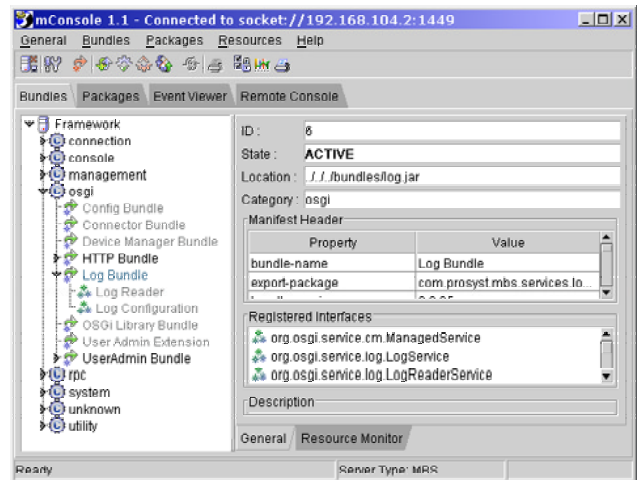
- The Control Unit tools ease the development, browsing and testing of control units as well as of UI applications for control unit management.
- The JSP Tool performs the translation of JSP pages into Java servlet classes.
- The Remote Console tool, suitable mostly for mobile phones, provides access to the OSGi framework when the used JVM has no console support.

Administration of the Framework

Developers can choose to manage the framework in a way that is most comfortable for them - through the visual interface of mConsole, or through text commands by using the framework local console or a standard Telnet client.

mConsole

The mConsole is a powerful application for local or remote visual administration of the framework – including bundle lifecycle management, configuration management of registered services, and events monitoring. It also provides some special features like log reading, user management, security management and more.



Fully featured administration with mConsole

Additional capabilities and packages of mBedded Server and other ProSyst products are explained in the data sheets available for each product.

If you have any further questions, we would be pleased to be of help. Please contact us at info@prosyst.com or visit our Developer Zone at dz.prosyst.com