
# Management

## Middleware

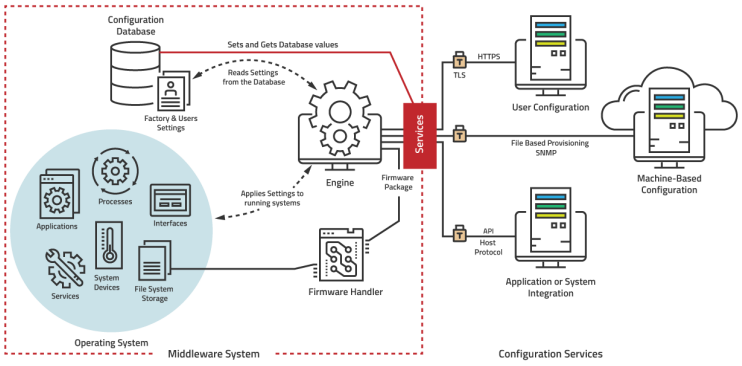
## overview

Arcturus Management Middleware is a device configuration and firmware management system for embedded Linux products.

The middleware is built around a central configuration database that stores factory and user settings. This database is coupled to a configuration engine that hooks existing OS subsystems and uses an internal set of policies to instantiate run-time configuration changes in a controlled way. Various higher-level methods are used to access the configuration database including a web user interface, REST, CLI, SNMP, XML and HTTPs “get” file-based provisioning. A set/get API is supported for application integration and a host protocol is available for compatibility with the Mbarx Secure IoT ecosystem.

Firmware management is handled using secure HTTPs “put” or “get” methods to obtain firmware packages. Metadata contained inside the firmware package defines the payload compatibility, file integrity and target filesystem location. The firmware management service can update one or more filesystem partitions or on-board bootloader. Extensions to the firmware management service can support redundant filesystem images by changing kernel parameters, after verifying a successful boot sequence.

A complete set of services are available to brand or customize any middleware component. Arcturus offers simple engagement packages to help get development moving quickly.




Embedded Linux Configuration and Firmware Management.

Arcturus
empower embedded.



**Arcturus Networks Inc.**

701 Evans Ave - Suite 300 Toronto, ON M9C 1A3
Tel: 1.866.733.8647 or 416.621.0125 Fax: 416.621.0190
arcsales@ArcturusNetworks.com
www.ArcturusNetworks.com

## overview

**Device Settings Database**

The database resides in persistent storage and uses kernel system calls to add, modify or delete database objects. Runtime configuration files for applications and services are built on-the-fly as part of the device initialization process, along with the configuration of interfaces. During runtime, the management engine automatically detects changes to the database, then processes the changes to instantiate them on the running system. Access to the database is typically provided through higher-level configuration methods; however, it is still useful to understand the architecture, which consists of:

- Berkley DB
- MIB-II database objects
- Factory default objects and values (e.g.: MAC address)
- User configured objects and values
- ASN1 object syntax
- Kernel API to set and get objects and values

**Configuration Engine**

The middleware engine is a daemonized service that is spawned during the Linux boot sequence and supervised by a controlling wrapper application. The role of the service is to processes changes made to the database using an internal system of policies and dependencies. The processing results in runtime configuration changes using the standard methods to stop and start Linux services and signal applications to reload configuration files. The management engine is a background process internal to the middleware that consists of:

- Initialization and supervision via application wrapper
- Database change detection
- Executable system of configuration policies and dependencies
- System logging

**Configuration Methods**

The configuration methods are a collection of tools that provide higher-level access to the configuration database. These methods are generally how a user or application would interact with the middleware system. These include:

- Web User Interface (HTTPs)
- Host Protocol (TLS)
- File-based remote provisioning (HTTPs)
- Interactive Voice Response (IVR) (voice products only)
- SNMP v1, v2, v2C
- REST (TLS)
- XML (TLS)
- Command line (typically for debug)

**Firmware Management**

The firmware management system ties into the configuration methods by providing a way to push or pull firmware updates. These updates can be uploaded by a user through the web user interface, obtained from a provisioning server automatically or can be managed at a system-level using the Mbarx Secure IoT tools. The firmware management system consists of:

- Firmware package with metadata
- Firmware package transports method (HTTPs)
- Firmware management daemon with filesystem awareness (including partitioning)
- Firmware package handler to check file integrity, payload compatibility and destination filesystem location
- Optional image redundancy and failover

## features

The middleware is fully integrated with common Linux services and interfaces allowing the configuration of network settings, system logging, access control, firewall, routing, VoIP, firmware management and administrative settings. The base platform is generally provided with a set of features and services representative of an embedded router platform:

- Linux Services Settings
- DHCP client / server
- Definable DHCP server address range
- DHCP client reservations
- Static IP address settings (netmask gateway, DNS)
- PPP / PPPoE
- Network bridging / bonding
- NTP client / server
- NAT firewall and port forwarding
- SPI and DOS protection
- Common ALGs
- Routing table and DMZ
- Local Hosting
- VLAN and TOS
- Telnet (disabled by default)
- SSH (disabled by default)

**Voice and Media Middleware Settings**

- Announce LAN address (IVR)
- Announce WAN address (IVR)
- Announce current UID (phone number) (IVR)
- VoIP account configuration
- Failover server configuration
- Calling modes
- Audio settings
- Codec selection
- Phone book
- Speed dials and push-to-call
- DTMF methods
- Incoming call handling
- Outgoing call blocking rules
- Signaling configuration
- Self tests and reporting
- Call quality reporting and handling
- Multicast PA paging

**Other Middleware Features**

- Admin login and password
- Mac address cloning
- Packet counters
- Interface statistics
- Ping and network tests
- Gateway discovery
- Diagnostic capture
- Save configuration
- Load configuration
- System logging
- Remote system logging
- Hardware watchdog service support
- CPU/thermal probe support
- Reboot
- Factory reset