Architecture

Table of Contents

Logical Design

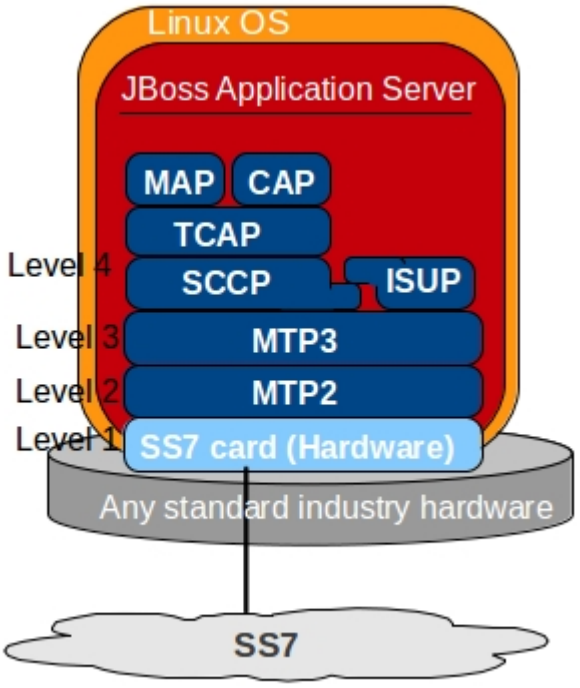
The Restcomm jSS7 is logically divided into two sections - lower and upper.

- The lower section provides implementation for SS7 Level 2 and Level 3. This section is therefore influenced by the type of SS7 hardware (Level 1) used.
- The upper section provides implementation for SS7 Level 4 and above.

This logical division offers great flexibility where hardware is concerned. Irrespective of the type of hardware used in the lower section, Restcomm jSS7 Level 4 and above remains the same. Since the API set is consistent regardless of the lower layers, you can easily migrate your applications from TDM equipments to M3UA. For example, applications using Restcomm SCCP stack (and/or upper layers) can easily be migrated from TDM equipments to Restcomm M3UA with just configuration changes and without changing a single line of code.

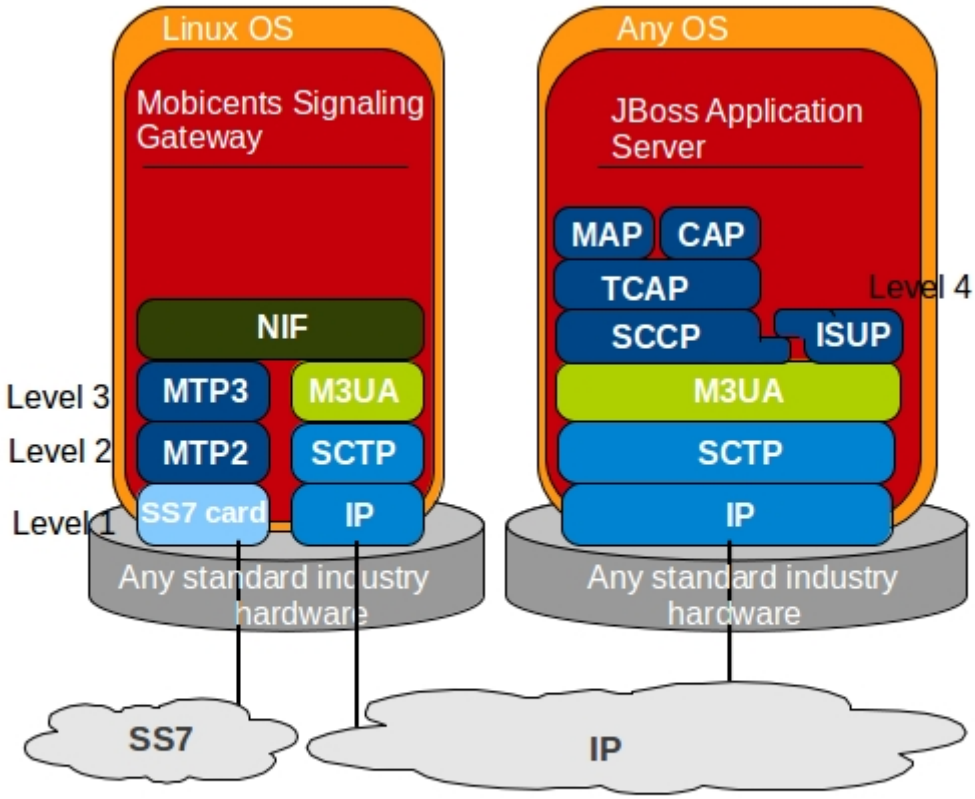
Restcomm jSS7 is designed efficiently to offer you the flexibility to install and use the Levels 2,3 and 4 in the same JVM and machine where SS7 Hardware (Level 1) is installed. Alternately, you can also install Level 1,2 and 3 in one machine and install Level 4 on a separate machine. In the second scenario, `M3UA` over `SCTP` is used for communication between Level 3 and Level 4 (on different machines) and this is referred to as Restcomm Signaling Gateway. The figures below illustrate the above 2 scenarios.

Scenario 1: The complete Restcomm jSS7 is installed in one machine.



The above two sce

Scenario 2: Restcomm Signaling Gateway - Level 3 and below are installed on one machine and Level 4 is installed on a different machine.



NIF - Nodal Interworking Function

If you use Restcomm M3UA stack, you must use JDK 7 to run the stack as well as to compile the source code. M3UA leverages Java SCTP which is available only in JDK 7.

For more details regarding installation, please refer to the Restcomm jSS7 Installation Guide.

Directory Structure

The top-level directory is named - and immediately below this are five sub-directories named *asn*, *_docs*, *oam*, *sctp* and *ss7*. All the functional modules of the Stack reside in the *ss7* folder.

```
|- restcomm-jss7-<version>
        |- asn

        |- docs

        |- oam

        |- sctp

        |- ss7
                |+ native
                |+ protocols
                |+ shell
                |+ restcomm-ss7-service
                |+ restcomm-ss7-sgw
                |+ restcomm-ss7-simulator
                |+ restcomm-ss7-traceparser
```

The following is a description of the important services and libraries that make up Restcomm jSS7

asn

Abstract Syntax Notation One (ASN.1) library is used by various Restcomm jSS7 protocols to encode/decode the structured data exchanged between Signaling Points over networks. For more details on the `asn` library, refer to the document included in the *_docs* folder. Applications using any of the Restcomm jSS7 User Protocols may never need to call an `asn` API directly, however it must be in the classpath since Restcomm jSS7 User Protocols refer to this library.

docs

All relevant documentation for Restcomm jSS7 .

oam

UI Management module

sctp

Stream Control Transmission Protocol (SCTP) Library provides the APIs over Java SCTP. This library will be used only if M3UA layer is used. For more details on the `sctp` library, refer to the documentation included in the *docs* folder.

ss7

This folder contains the core protocol libraries that will be used by end applications as well as by the SS7 Service deployed in JBoss AS. The sub-directories included in the *ss7* folder are:

- *restcomm-ss7-sgw* : Standalone Signaling Gateway
- *restcomm-ss7-service* : SS7 service is the core engine and is used in conjunction with JBoss AS. The installation guide will teach you to install the Stack as a standalone component if you do not wish to run it as a JBoss AS Service.
- *restcomm-ss7-simulator* : SS7 Simulator is an application for testing the SS7 stack and

displaying its functionality. It is also a good example of how to use this stack.

- *restcomm-ss7-traceparser* : mobicents jSS7 Stack Trace Parser is a simple tool that can parse trace capture data files (of some formats) and log its content in some log file in a text form
- *native* : This folder contains the native libraries to interact with the SS7 Cards installed on the server. As of now, native libraries are compiled and available only for linux OS.
- *protocols* : Restcomm jSS7 User Protocols libraries. Your application will directly call the APIs exposed by these libraries. Depending on the application, you may be interested in either `TCAP` or `MAP`, or both, or `ISUP` libraries.
- *shell* : This holds the Command Line Interface (CLI) module to manage the Restcomm jSS7 .

Functional Blocks

The major functional modules of the jSS7 are:

1. SS7 Service *[dir: restcomm-ss7-service]*
2. Signaling Gateway *[dir: restcomm-ss7-sgw]*
3. Shell *[dir: shell]*
4. GUI *[dir: ui]*
5. SS7 Simulator *restcomm-ss7-simulator]*

The following sub-sections discuss in detail about the functionality of these individual components.

SS7 Service

SS7 Service creates an instance of higher layer of the Restcomm Stack and binds the instance to JNDI. SS7 Service is a JMX based service deployed in JBoss Application Server. It hides the underlying details like whether Level 4 and above are connected to peer via `M3UA` or if connected to the SS7 Hardware installed in the same machine as Level 4.

Following services are bound:

Table 1. SS7 Services

Stack Name	JNDI Name	Comments
TCAP	java:/mobicents/ss7/tcap	Exposes TCAP Stack via JNDI
MAP	java:/mobicents/ss7/map Exposes	MAP Stack via JNDI
CAP	java:/mobicents/ss7/cap Exposes	CAP Stack via JNDI
ISUP	java:/mobicents/ss7/isup	Exposes ISUP stack via JNDI

The figure below depicts the elements that are deployed as part of the SS7 MAP Service.

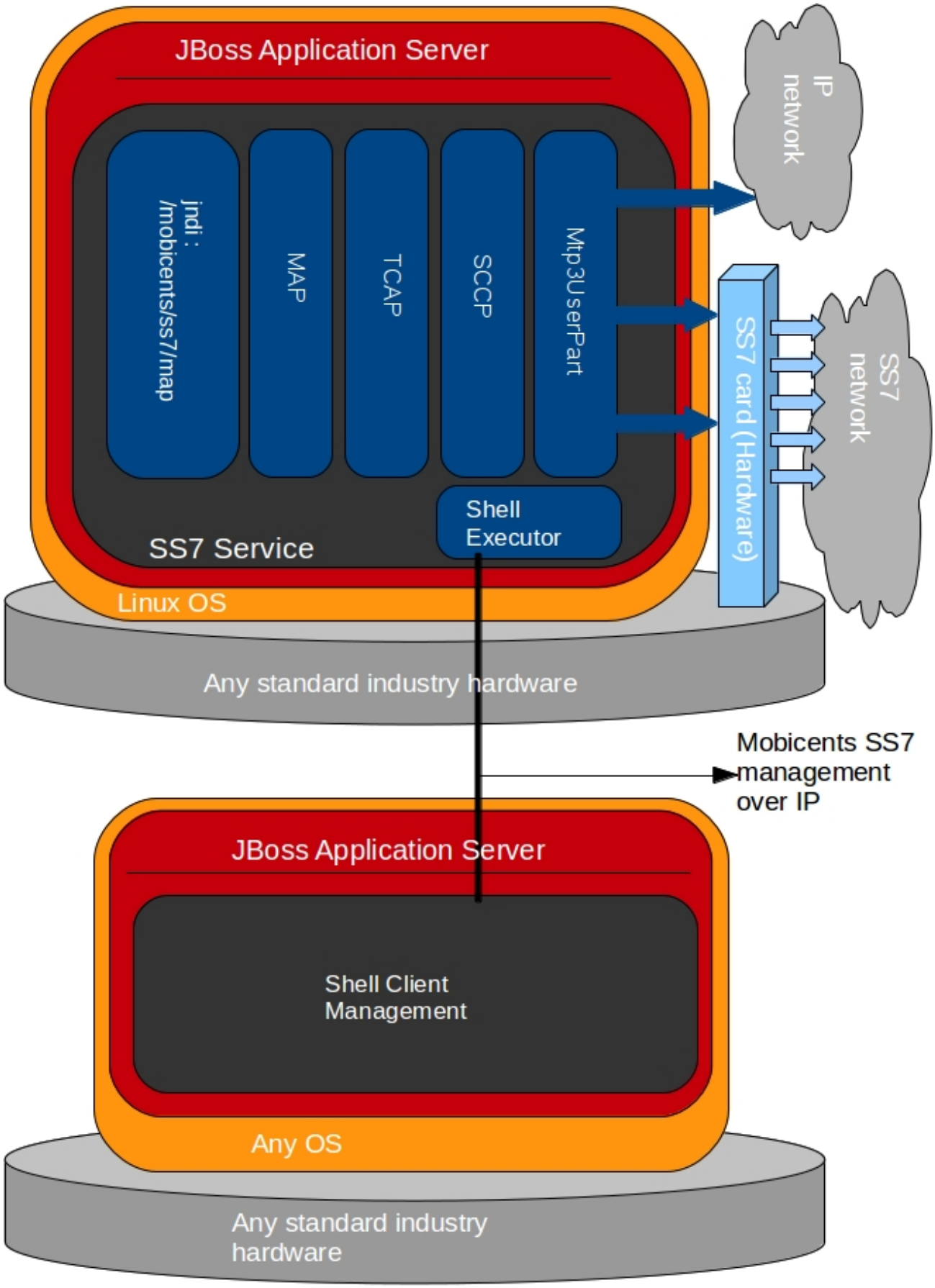


Figure 1. Restcomm jSS7 Stack Service Elements

SS7 Service Elements serve the following purposes:

Expose protocol access points

Access points allow users to access lower layer protocols like `MAP` and interact with the SS7 network through such protocols.

Expose management interface

`Shell Executor` allows the `Shell` client to connect and issue commands.

The figure below depicts the elements that are deployed as part of SS7 Service.

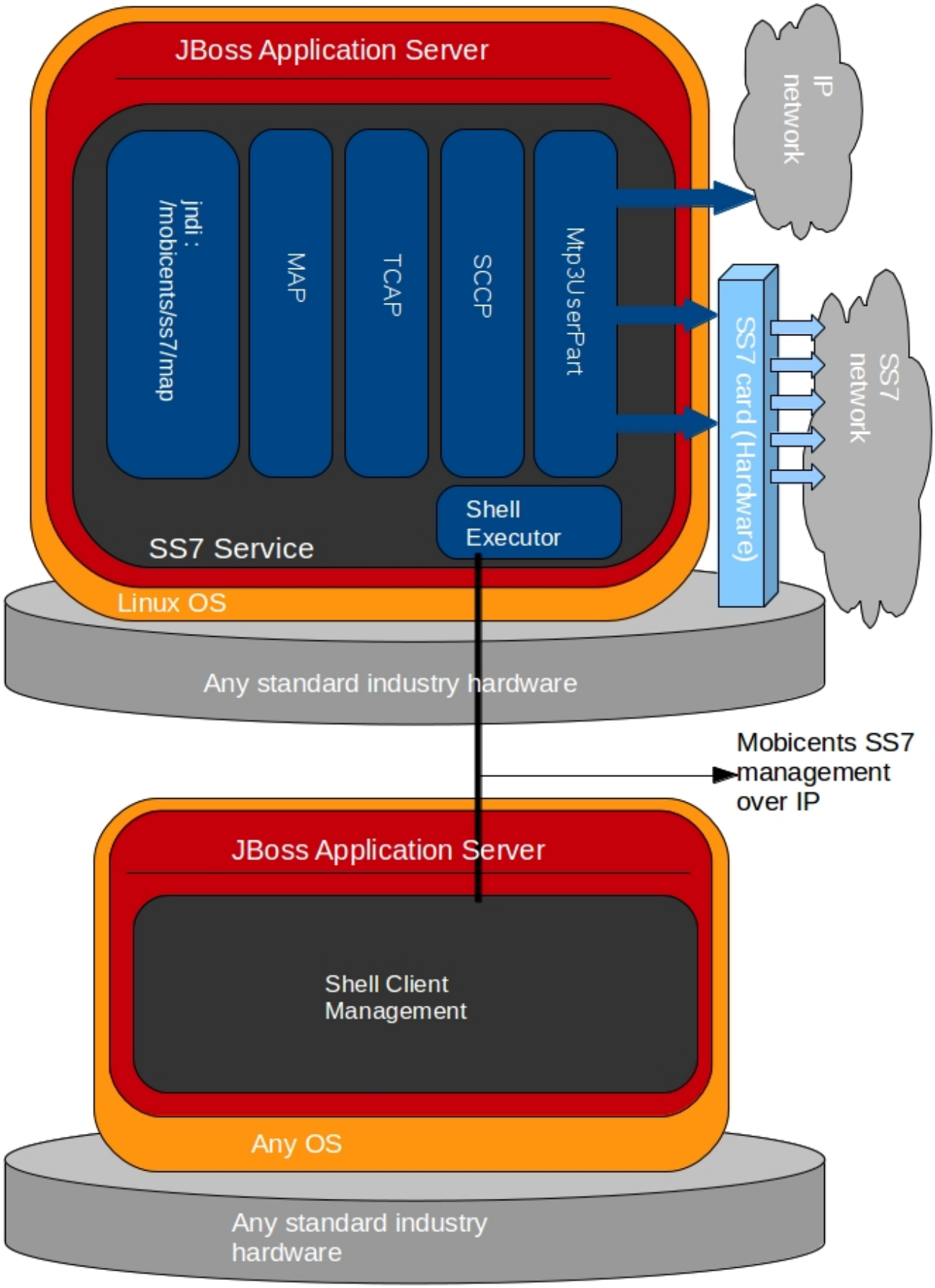


Figure 2. Restcomm jSS7 Stack Service Elements

For more details on Running and Configuring the SS7 Service Elements, please refer to the chapter [_running_jss7].

Stack Usage

The figure below depicts how Restcomm jSS7 is used.

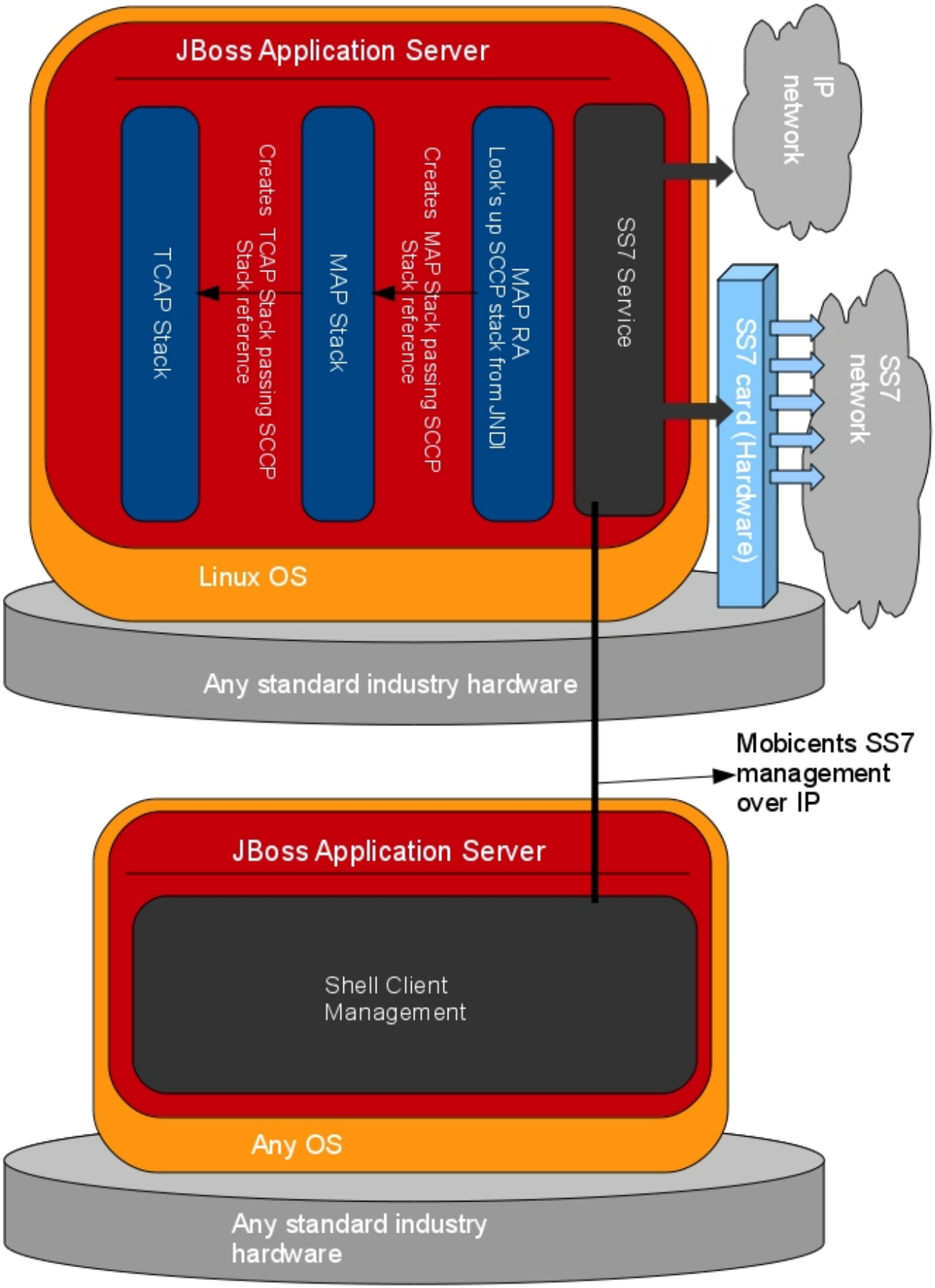


Figure 3. Restcomm jSS7 Stack General Design

Signaling Gateway

Restcomm Signaling Gateway (SG) is a signaling agent that receives and sends Switched Circuit Network (SCN) native signaling at the edge of the IP network. Restcomm Signaling Gateway leverages MTP and Restcomm `M3UA` Stack explained in [_mobicents_signaling_gateway_m3ua].

The figure below shows the components included in Restcomm Signaling Gateway. Configuring the Signaling Gateway is explained in the chapter [_running_jss7].

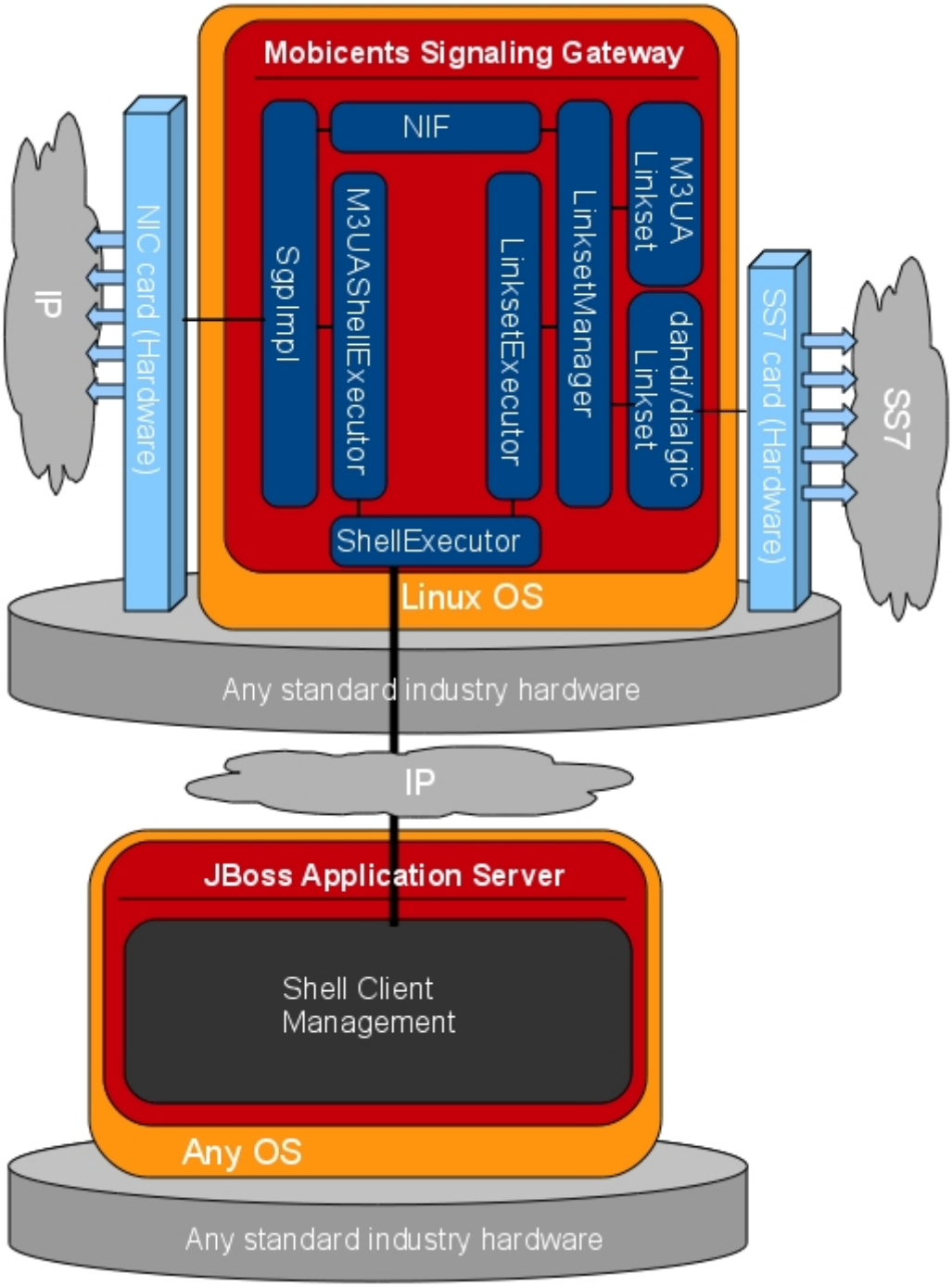


Figure 4. Restcomm Signaling Gateway Components

Shell - Comman Line Interface

`Shell` is a Command Line Interface (CLI) tool that will allow you to manage different aspects of Restcomm jSS7 in an interactive manner. It connects to different instances of Restcomm jSS7 which manages `Linksets`, `SCCP` resource, routing and `M3UA`.Usually `Shell` will be invoked from a remote machine(remote to `Linksets` and application protocols).

Graphical User Interface

The Graphical User Interface will allow you to manage different aspects of Restcomm jSS7 through a convenient user-friendly interface. You can launch the GUI in any Web Browser and manage the Stack instance efficiently using the GUI operations.

SS7 Simulator

Restcomm jSS7 comes with a Simulator that will help you to understand the functionality of the Stack. The Simulator may be used as an application for testing the SS7 Stack or as an example of how to use this Stack. You can run several instances of the Simulator in a single machine and each instance can have its own configuration. In addition, the Simulator offers you the flexibility to run it locally or remotely. You must remember to configure all layers before running tests with the Simulator.

The Simulator contains three layers of SS7 stack components and one testing task layer which presents the concrete testing task. You can select from these layers as required, however some layers demand corresponding lower layers. For example, the `TCAP+MAP` layer demands `SCCP` as layer 2. Depending on your testing goals, you can configure each of these layers separately but the configuration options are limited and do not cover all possible SS7 Stack options.

Simulator Layers

1. Layer 1 [MTP3]
    - M3UA
    - DialogicCard
    - DahdiCard [yet to be implemented]
2. Layer 2
    - SCCP
    - ISUP [yet to be implemented]
3. Layer 3
    - TCAP + MAP
    - TCAP + CAP
    - TCAP + INAP [yet to be implemented]
4. Testing Task Layer

- USSD client test
- USSD server test
- SMS server test
- SMS client test
- CAP SSF test
- CAP SCF test
- MAP ATI client test
- MAP ATI server test