Applicata **BRINGS VALUE NOT COSTS**

JN Signalling Processing Unit
An Overview

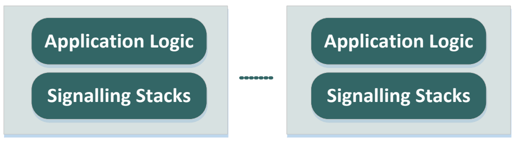
Easy integration
High Scalability and Availability
Signalling Network Function Virtualisation
Integrated Protocol Stacks, Configuration and Monitoring

Signalling Architectures, Separation of Signalling and Application Planes

Services and applications running in GSM, IMS and EPC networks require signalling protocol stacks. Embedding the stacks in the same process where the application runs is a solution that follows a non-distributed architecture.

A non-distributed architecture where the signalling and application planes are not separated has disadvantages, including limited scalability and performance capabilities, and higher costs in case of scaling by multiplying protocol layer licenses.

Signalling and application planes are separated in distributed architectures. Signalling plane can be shared between the applications running at application plane.



Non-Distributed Architecture:
Signalling and Application modules share the same process.
Scaling requires separate signalling modules and licenses.

Inside this publication:

Applicata Signalling Processing Unit

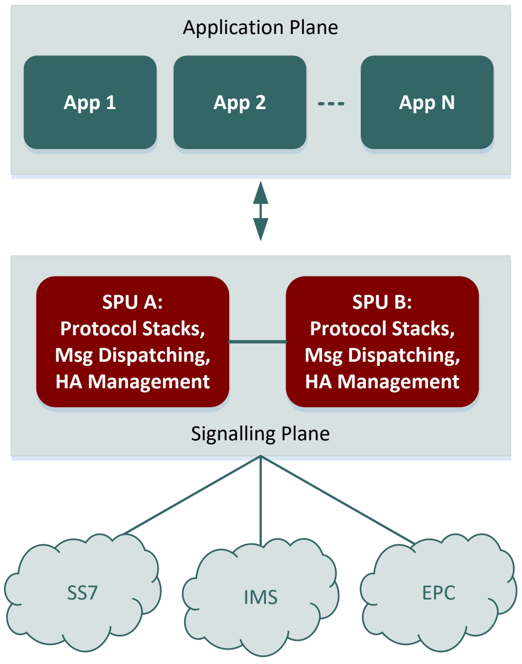
Applicata JN Signalling Processing Unit (SPU) is a signalling layer software-only implementation covering SS7, SIP[†] and DIAMETER protocols.

It implements the separation of signalling and application processes in a distributed architecture. No specialised hardware is required and SPU functions can be virtualised.

The signalling stacks and the application(s) run in different processes communicating with each other over IP.

All applications and application instances at the application plane share the same Signalling Processing Unit(s) running at the signalling plane. The applications can be easily scaled without affecting the signalling layer. Typically, two SPUs are enough at the signalling layer, providing dual resiliency and high availability.

JN SPU comes with Signalling Layer Interface libraries for different programming languages. These hide the complexity of handling the interface with the SPU, convert the protocol messages to/from language specific structures, and make the integration of applications very easy. For example, the applications implemented in Java, C/C++, Go, Erlang, Python or other languages, including scripts, can use JN SPU for signalling. Similarly, the SPU interface can be implemented by Resource Adapters running in JAIN SLEE containers.



Application Plane
App 1
App 2

App N
SPU A: Protocol Stacks, Msg Dispatching, HA Management
SPU B: Protocol Stacks, Msg Dispatching, HA Management
Signalling Plane
SS7
IMS
EPC

Applicata Signalling Processing Unit:
Signalling and Applications planes are separated in a distributed signalling architecture.
Applications can easily scale sharing the same signalling modules and licenses.

KEY POINTS

- **Easy integration with applications written in different languages**
- **Reduced costs of ownership**
- **Proven and extendable platform**
- **Extremely high performance and availability**
- **Round the clock support**

Applicata JN SPU offers a cost effective and feature reach signalling solution with integrated protocol stacks, configuration and monitoring, rate control, message dispatching, very high performance, easy integration, network virtualisation, service high availability and scalability.

[†] In the roadmap

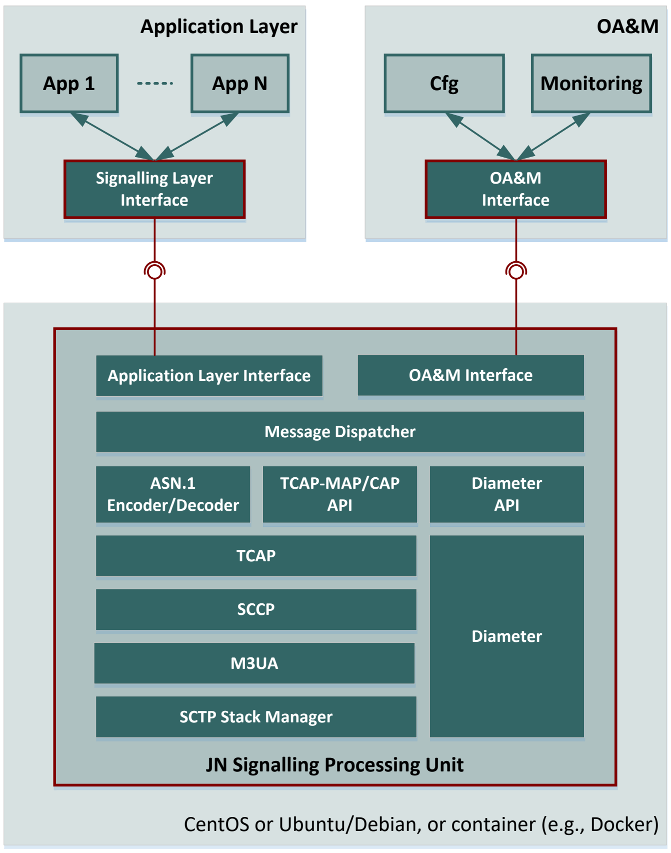
Signalling Processing Unit Modules

JN SPU contains the following software modules:

- SS7 protocol stack modules, including M3UA, SCCP, TCAP, MAP, CAP,
- Diameter modules, including Diameter protocol stacks and application dictionaries,
- ASN.1 Encoding/Decoding modules for full support of MAP version 1,2,3 and 4 and CAP Phases 1,2,3 and 4 according to 3GPP specifications
- MAP, CAP and Diameter API modules
- Message dispatcher module
- Interface modules to Application layer and Operation, Administration and Management

JN SPU software is available as RPM package for L:inux RedHat/CentOS distributions, DEB packages for Ubuntu Trusty and Xenial distributions. It can be also delivered as a fully equipped Docker container containing the operating system and SPU software. modules.

All JN SPU distributions are purely software based. JN SPU package or container can be installed on physical servers or virtual servers, including clouds.



Application Layer Interface

JN SPU Application Layer Interface (ALI) provides message based API for the application layer. Messages over ALI between the SPU and Application plane are exchanged over TCP/IP formatted according the Erlang External Term Format (ETF), http://erlang.org/doc/apps/erts/erl_ext_dist.html.

The API over this interface includes management messages and protocol specific messages.

JN SPU ALI module is responsible for formatting and serializing the messages to Application plane in ETF format and deserializing the messages received from the Application plane.

The management messages over ALI provide functionality for the applications to register with the SPU node(s) and to exchange their capabilities.

Application capabilities sent to SPU include some protocol specific parameters. For example, MAP specific capabilities may include the values or ranges of IMSI, MSISDN etc. parameters; CAP specific capabilities may include the values or ranges of Service Key and/or IMSI parameters; Diameter specific capabilities are defined as sets of values of User-Name and Realm AVPs.

SPU Message Dispatcher module uses the information from the capability exchange to route the incoming messages to the appropriate application.

The protocol messages over ALI carry the protocol specific operations and parameters.

MAP and CAP protocol messages include dialogue handling for opening, continuing and closing dialogues, and service specific messages carrying decoded TCAP components.

Diameter protocol messages follow Diameter protocol requests and responses.

JN SPU can be easily integrated with applications written in different, including scripting, languages

Signalling Layer Interface

JN SPU comes with a set of language specific Signalling Layer Interface (SLI) libraries for use by the applications at the Application layer.

SLI libraries include functionality for:

- establishing TCP/IP communication with SPU node(s),
- deserializing/serialising the messages from/to SPU nodes(s), and
- providing messages and message parameters via callbacks to the applications.

Using SLI libraries the application(s) can establish connection with a single SPU node or with several SPU nodes that may be connected to the same or to different networks.

SLI libraries remove the need for implementing the functionality for parsing the data received from SPU in binary ETF format at the application side, and serializing it when data is sent to SPU nodes.

When ETF buffer is parsed, the language specific SLI library calls callbacks provided by the application for each parsed message parameter and supplies the corresponding tag and value. This way the application can easily convert the message from ETF format into a language/platform specific structure.

SLI libraries radically facilitate the SPU integration with applications implemented in different languages. Currently, SLI libraries are available for Java, Go, Python and Erlang. SLI libraries for other languages (e.g., C/C++, Javascript etc.) can be delivered on request.

Following the same approach Java SLI library can also be wrapped into JAIN SLEE Resource Adapters.

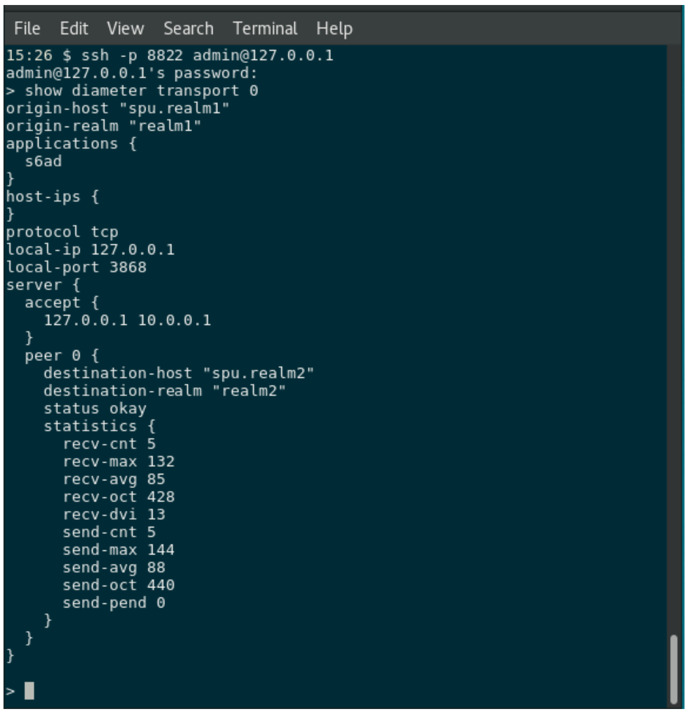
OA&M Interface

JN SPU contains a lightful and powerful custom SSH shell implementation that provides functionality for secure configuration and monitoring.

The configuration and monitoring commands can be invoked using a standard SSH client console. JN SPU provides user access control over SSH based on password or public/private keys.

The configuration and monitoring functionality is based on a management model of the SPU system that represents the configurable and monitorable parameters in a tree view. The custom shell implements commands to view and/or edit the branches or leaves in the tree.

The execution of the commands sent over the OA&M Interface at JN SPU side is very efficient. Unlike the standard SSH daemon in Linux the OA&M command execution in JN SPU does not require starting a new bash shell process in the operating system on each SSH connection. Instead, JN SPU SSH shell is implemented in Erlang and new SSH connections require starting of a Erlang process, an operation that is extremely light. This efficiency is especially important for handling the monitoring requests sent periodically and, possibly, frequently by monitoring consoles that typically use a pull based monitoring mechanism.



```
15:26 $ ssh -p 8822 admin@127.0.0.1
admin@127.0.0.1's password:
> show diameter transport 0
origin-host "spu.realm1"
origin-realm "realm1"
applications {
  s6ad
}
host-ips {
}
protocol tcp
local-ip 127.0.0.1
local-port 3868
server {
  accept {
    127.0.0.1 10.0.0.1
  }
  peer 0 {
    destination-host "spu.realm2"
    destination-realm "realm2"
    status okay
    statistics {
      recv-cnt 5
      recv-max 132
      recv-avg 85
      recv-oct 428
      recv-dvi 13
      send-cnt 5
      send-max 144
      send-avg 88
      send-oct 440
      send-pend 0
    }
  }
}

> 
```

Configurations and Scalability

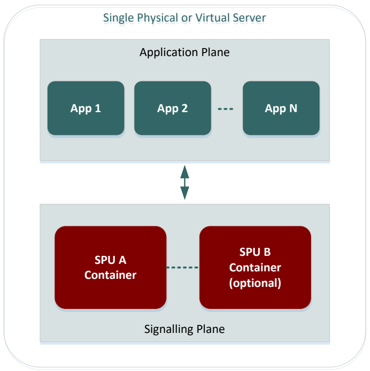
JN SPU can be deployed in different configurations depending on the requirements. The smallest configuration comprises one or two instances of SPU sharing the same physical or virtual machine with the application(s).

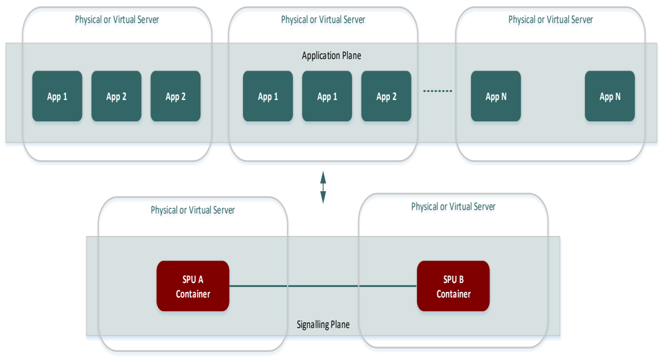
The signalling layer still remains separated from the application layer. Multiple applications share the same SPU(s).

the operation.

SPUs and applications can be deployed on multiple physical or virtual machines. This configurations provides high availability and the highest scalability and performance.

Signalling layer licensing costs do not increase when the system scales up by adding additional applications, application instances and/or application servers.




Physical or Virtual Server
Physical or Virtual Server
Physical or Virtual Server
Application Plane
App 1
App 2
App 2
App 1
App 1
App 2
App N
App N
Physical or Virtual Server
Physical or Virtual Server
SPU A Container
SPU B Container
Signalling Plane

Easy Application Logic Integration

JN SPU modules are implemented in C and Erlang. Its implementation is capsulated and it runs in separate processes. JN SPU instances can even run on separate machines.

The language and the platform used for JN SPU implementation does not imply any restrictions regarding the programming languages and platforms chosen for the applications. Accordingly, the application logic can be implemented using the preferred language or platform, for example Java, C# or C/C++, and different services/applications may use different languages.

In addition, JN SPU interface can be used by application logic written in some widely spread scripting languages, for example Javascript or Python, providing an option that may significantly shorten the service development and deployment time.

High Availability

JN SPU supports resilient configurations where more than one SPU instance is run in active-active mode of operation.

SPU instances run in a cluster load-sharing the signalling traffic between themselves. If one of the instances fails the surviving instances can continue the operation.

Applicata Signalling Processing Unit At Glance

Protocols

- MAP versions 1,2,3 and 4
- CAMEL Phase 1,2,3 and 4
- SIP†
- DIAMETER
- SSH for OA&M and provisioning

Features

- SIGTRAN M3UA peer-to-peer and ASP-SGW connectivity
- Multiple local SS7 point codes
- Full support for SS7 MAP and CAP
- SIP† and DIAMETER support
- SS7 MAP, CAP, SIP† and DIAMETER API for the applications
- Fully software based
- Packaged as a rpm or deb or container for quick and easy deployment

Benefits

- Easy integration with applications written in different languages
- Investment protection: signalling layer can be shared between different applications/services
- Open scalability and high performance
- Multi-server configuration option
- Commercial licenses do not restrict the capacity per server
- Network Virtualised Function option
- Cloud based operation option

www.applicata.bg

About Applicata

Applicata specialises in the design, development, installation and integration of systems and software for law enforcement and telecommunications. Applicata team guarantees that top quality products and services will be delivered within tough deadlines and budgets. Applicata is well known by its competitive advantages, including low cost, flexibility in covering specific customer needs, on time delivery and extended support.

Sales Europe	Sales Asia	Applicata
Ashwood, Woodcote	10F, No. 1, Lane 25, XinSheng S. Rd., Sec. 3	6, Angista Str
Guildford, Surrey, GU2 4HQ, UK	Taipei 10660, Taiwan R.O.C.	1527 Sofia, Bulgaria
Tel: +44 1483 506384	Tel: +886 2 8369 5685	Phone: +359 2 981 4203
Fax: +44 1483 506384	Fax: +886 2 8369 5685	Fax: +359 2 943 4719
sales@applicata.bg	sales@applicata.bg	office@applicata.bg
www.applicata.bg	www.applicata.bg	www.applicata.bg

† In the roadmap