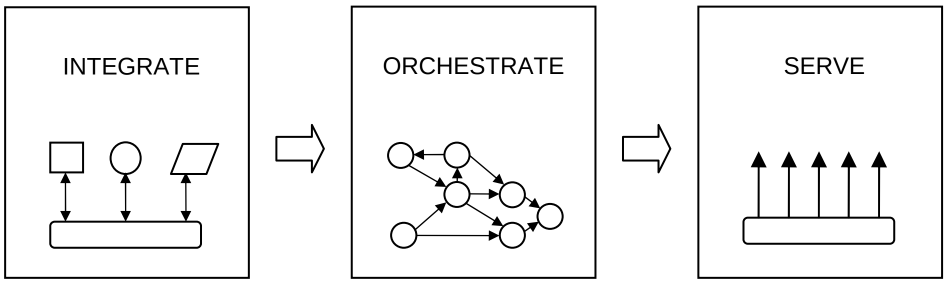
Nirva Application Platform overview

Document version: 1.09

Table of Contents

What is the Nirva Application Platform?

Nirva Application Platform is a robust application server combined with integration and orchestration services.

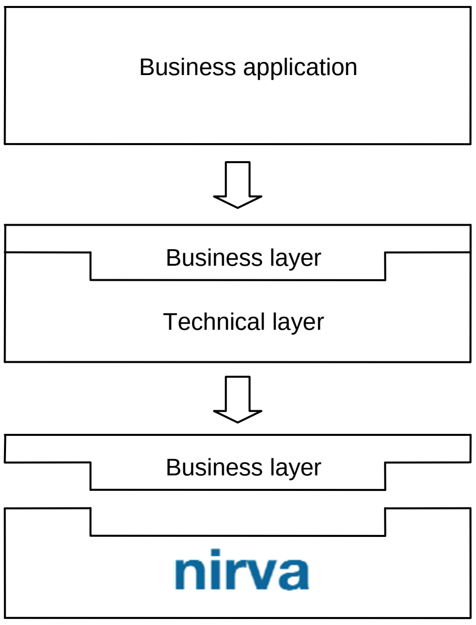


Application server

All enterprise applications or software products have two distinct parts: The business or functional part and an underlying infrastructure bearing the responsibility of managing all the technical aspects: communication with clients and systems, data merging between different sources, scalability, security, integration, session management, Web user interface, component reusability, etc.

The latter is also the main role of an enterprise application server providing a robust and high performance technical platform.

Architects and developers can then concentrate their efforts on building business related functionality while reducing their reliance on the underlying technical aspects.



Nirva Application Platform provides the necessary infrastructure to host enterprise applications and stand alone products alike.

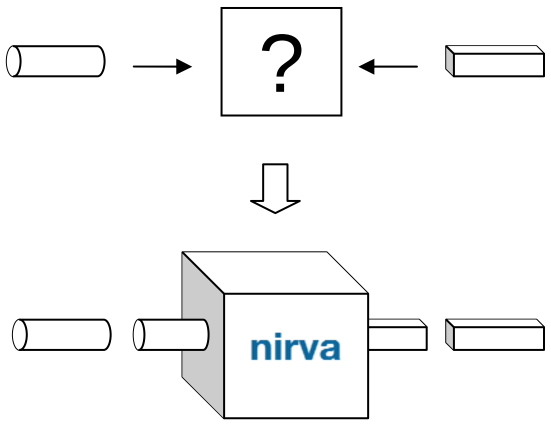
Integration services

The Nirva Application Platform integration part, although quite capable of being implemented as a strategic component, often remains perceived as primarily tactical as typical similar requirements are frequently met on the market.

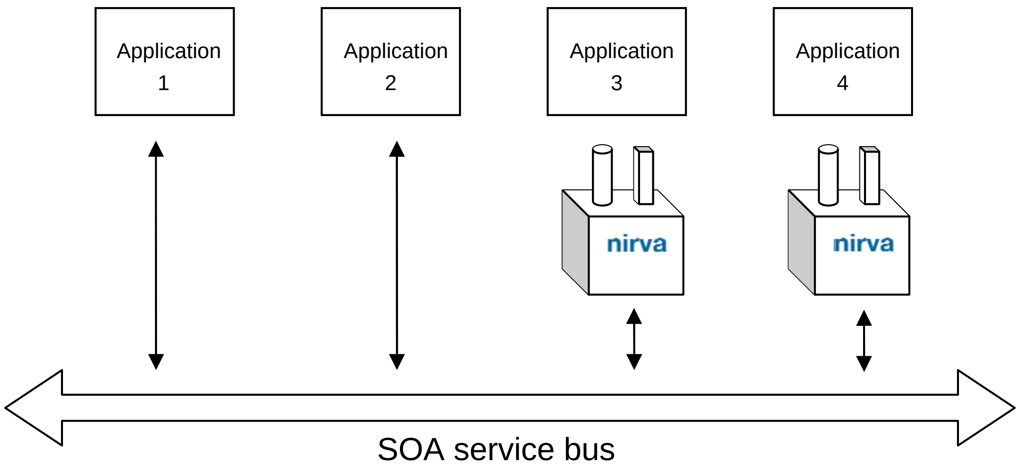
Newer strategic products, based on SOA and Web Services are often too remote from the technical reality to provide a suitable tactical answer for a quick business solution turnaround. In fact such products integrate applications already prepared to communicate via Web Services.

On the other hand, Nirva Application Platform is able to integrate not only Web Services but also application APIs seen as pieces of code or technology to access the business functionality offered by core systems. For example Nirva Application Platform can erase boundaries between Java and .Net by allowing these two worlds to coexist in a single process where it can communicate between them.

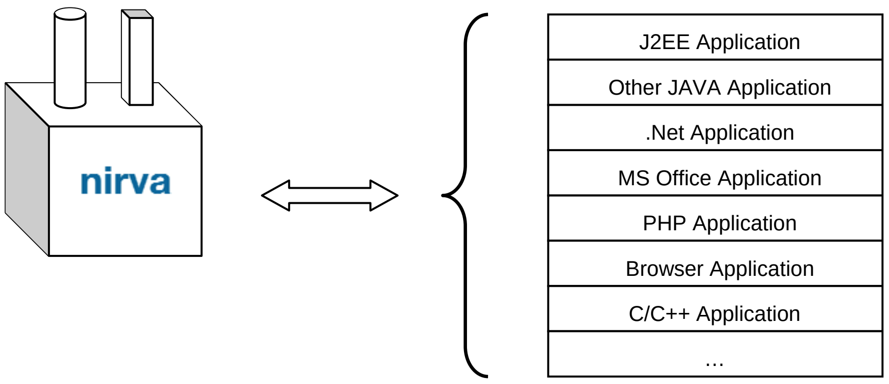
Nirva Application Platform can be positioned as “the part and the glue that allows a seamless connection between a round peg and a square hole”.



As a tactical integration product, Nirva Application Platform can be used in conjunction with other SOA products to “Web Service” entire applications or any of their components:



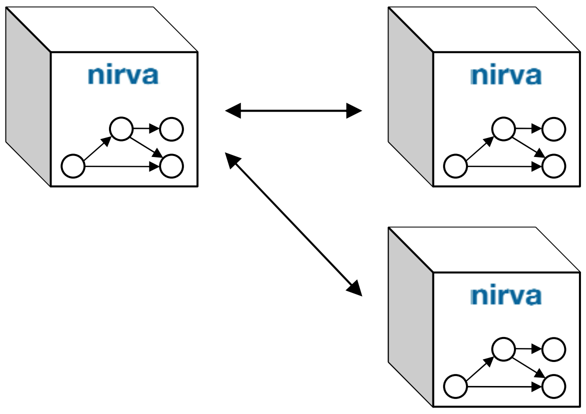
Nirva Application Platform does not stop at offering Web Services connectivity. Its architecture allows for instance the publishing of a part of a hosted application to third party components through a vast array of technical connectors:



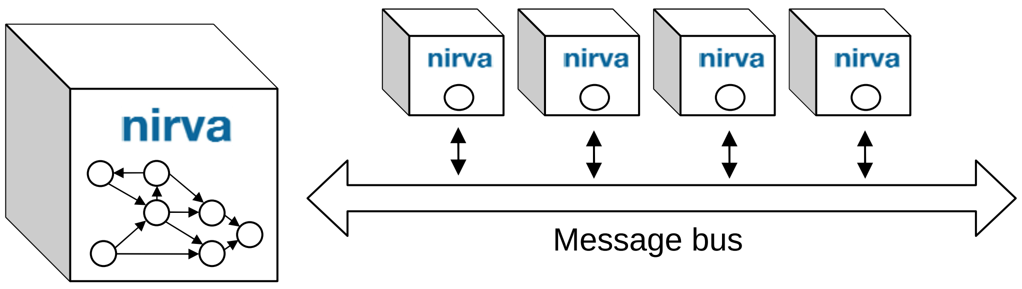
Orchestration

Nirva Application Platform provides orchestration features that help in building applications with a business perspective in mind.

Orchestration can be coded into simple procedures written in known languages (Java, Perl, .Net) and distributed across several servers.



Orchestration can also be controlled by a dedicated workflow service that routes activities to agents across a message bus. This mode facilitates the scalability of the system and gives a clear business view of the process.



Who is Nirva Application Platform for?

Nirva Application Platform is used as:

- An Application Server for enterprise-wide projects.
- An Application Platform for the products.

Projects

Nirva Application Platform targets tactical projects for large and medium companies, especially for back-end processes. This kind of project is often motivated by cost reduction but is paradoxically supported by oversized products (e.g. WebSphere, WebLogic). Nirva Application Platform allows significant cost savings, not only for licenses, but also for development, maintenance and operations costs.

For the larger companies, Nirva Application Platform adequately complements the enterprise strategic tools by allowing them to be perceived as a business solution by hiding the technical aspects of the various components and by federating heterogeneous information and application systems.

For medium size companies, Nirva Application Platform projects can be used as a more strategic tool by supplying a robust, independent and complete application base layer.

Products

For software suppliers, Nirva Application Platform is used as an application backbone supporting their own products. Its feature rich and flexible architecture hides a significant part of the technical components, thus allowing product development to focus on core functionality. Nirva Application Platform offers value from the technical integration layer up to the presentation and front-end layer.

Nirva Application Platform supports the development of applications opened to the outside world (APIs), secured (users and rights management) and extensible (distributed architecture). Its application domain is unrestricted (bank, insurance, industry, health care, etc.).

Nirva Application Platform can host and support standard applications (client/server) or «Software as a Service» (SaaS) type with multi tenancy management.

Users are small and medium software suppliers. Nirva Application Platform can also be used by specialized integrators who wish to standardize and reuse their development.

How Nirva Application Platform works

Nirva Application Platform is a standalone multithread engine containing all the necessary technology to host classic or Web applications.

All-in-One concept

Nirva Application Platform is a self-sufficient, all-in-one product. All necessary components (with the exception of traditional databases) to manage applications are supplied within the product:

- Web Server
- HTTP/HTTPS Server
- Integrated Java environment
- Integrated Perl interpreter
- Embedded .Net CLR engine (windows only)
- XSLT processor
- Scheduler
- Listeners
- Workflow
- Deployment tools
- Test suite
- Monitoring tools
- Licenses creation and maintenance tool
- Web Services (producer and consumer)
- Application container
- Upward (services) and downward (clients) connectors
- Security services
- SQLite file database

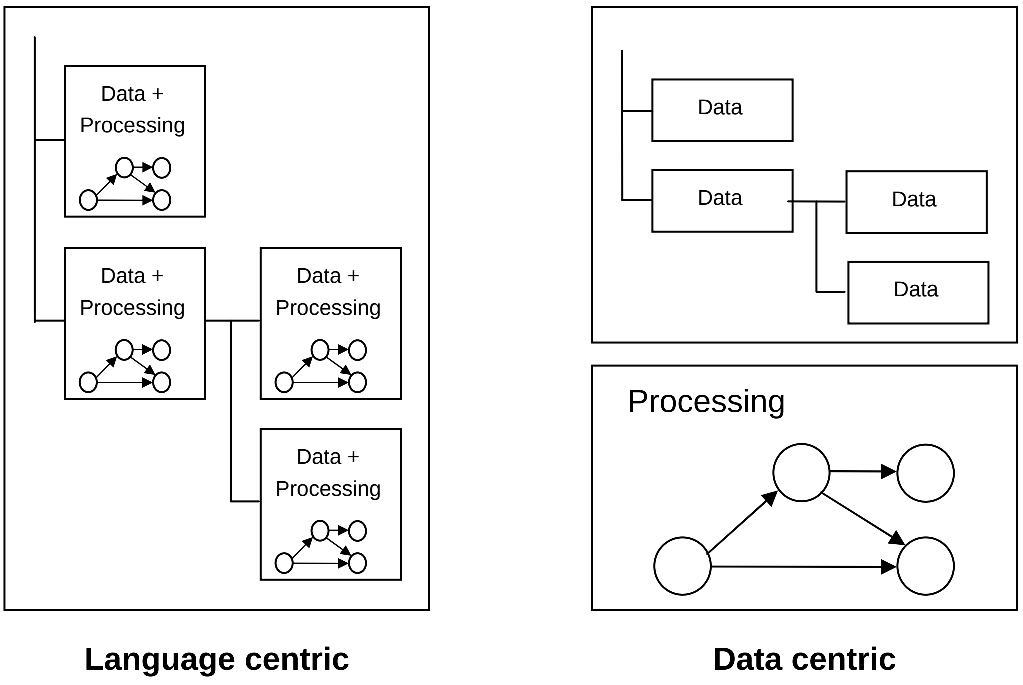
Even if Nirva Application Platform delivers all these components, some of them can be externalized. For example, the Java virtual machine or the XSLT processor.

This all-in-one concept caters for significant development and running cost reductions.

Data centric architecture

The architecture is data centric as opposed to language centric, which is often the case with most traditional application servers (Java or .Net). This particular point is at the core of the product originality.
Data are organized in hierarchical containers that can be persistent or not, shared or specific to user sessions. The container data comes from user data, databases or other processing.

Coding is only used to process and organize this data that can then be presented to the user.



This model allows for loose coupling between the data and the code that manipulates them. For example, it is possible to use different programming languages for a single application. Nirva Application Platform uses Java, .Net (C#, Visual Basic), Perl and C++ as processing languages. Each language's strength can be exploited and existing code can be reused.

Main components

Thanks to the simplicity of its architecture (containers, services, procedures, applications, commands and connectors) Nirva Application Platform focuses on business solutions and reduces development time.

Nirva Application Platform enables loose coupling between its various components to enhance product evolution.

These components are:

- ***Container***. Stores business data at user context level (session) or at application level (shared data). The hierarchical container stores well defined objects (boolean, integer, string, string list, indexed string list, table, file, and binary data) and can also store sub-containers. Containers can be persistent in registries or in a mass storage service.
- ***Command***. A command is a simple string of characters containing the name of a service, the name of the command and a set of parameters in the form of name=value. A command retrieves data from a container and populates another one. This loose coupling allows unmatched flexibility in the product.
- ***Service***. A service is a set of commands for a particular domain. There is a SYSTEM service fully integrated in Nirva Application Platform that manages container data access and other system wide features. Users can also create their own services to add functionality to the product. Services can be programmed using C++, Java or .Net. Services can send commands to other services. A Service is a documented, configurable and deployable component with a separate life cycle. A Service can be used by several applications.

- ***Procedure***. A procedure is a portion of code that chains commands according to business logic. They can be written in Perl, Java, .Net or a simple text file. Each command can execute one or several procedures written in different languages.
- ***Application***. An application is composed of a set of procedures, a Web site and the files needed for the presentation layer. All the application components are stored in a dedicated directory.
- ***Connectors***. Connectors are used to access application functionality from external applications or from a Web browser.

Key features

- **All-in-One concept**. Nirva Application Platform is a self-sufficient, all-in-one product. All necessary components (with the exception of databases) to manage applications are supplied within the product.
  Learn more…
- **Data centric architecture**. The architecture is data centric as opposed to language centric. Data are organized in hierarchical containers that can be persistent or not, shared or specific to user sessions.
  Learn more…
- **Simple model**. Thanks to the simplicity of its architecture (containers, services, procedures, applications, commands and connectors) Nirva Application Platform focuses on business solutions and reduces development time.
  Learn more…
- **Multi language**. Nirva Application Platform applications and services can be developed in the following languages: Perl, Java, .Net (C#, Visual Basic), C++. Languages can be mixed.
  Learn more…
- **Scalable distributed architecture**. Nirva Application Platform is built as a distributed architecture to support application loads. It provides failover and load balancing features.
  Learn more…
- **Standards-based**. Nirva Application Platform is based on well defined standards.
  Learn more…
- **Extensible**. The product functionality can easily be extended thanks to the service component. The Nirva Application Platform service is a transverse component with its own life cycle and can be used to add new commands to the system.
  Learn more…
- **Component reusability**. Components can be reused to share the common part of applications. This boosts application agility and significantly reduces conception and maintenance costs.
  Learn more…
- **Upwards and downwards integration**. Nirva Application Platform can integrate components, applications or external technology to build composite applications (upwards integration) but can also integrate these applications in other applications with a set of connectors (downwards integration).
  Learn more…

- **Message bus**. Nirva Application Platform contains a message bus to support process management in asynchronous mode. In this mode, a central entity stores messages (activities) in queues where specialized agents can pick them up for processing.
  Learn more…
- **Session management**. Nirva Application Platform supports the concept of a session that represents a user's context.
  Learn more…
- **Security**. Nirva Application Platform supplies security functions as a standard. The security model is based on permissions, roles and users. Security includes SSO (Single Sign On management) to allow automated authentication of users already connected on a particular domain.
  Learn more…
- **Web Services**. Nirva Application Platform is both a producer and consumer of Web Services. Web Service creation is extremely simplified. A Web interface is used to define the structure of the Nirva Application Platform containers for input and output messages and to define a procedure to execute the relevant Web Service. Nirva Application Platform automatically generates the WSDL code describing the Web Service.
  Learn more…
- **Scheduler**. A scheduler is integrated in the core system and supports the planning of various tasks during a day, a week, a month or a year.
  Learn more…
- **Listeners**. Listeners wait for events. They can be used for example to watch a directory to retrieve files and to store them in the application for processing.
  Learn more…
- **Presentation layer**. Nirva Application Platform applications can supply a presentation layer accessible through a standard Web browser. The presentation layer works with HTML renderers (e.g. XSLT or JSP) that transform the content of an output container to HTML flow. Alternatively, Nirva Application Platform provides a framework based on reusable and customizable JavaScript components named Nidgets.
  Learn more…
- **Workflow**. The Workflow service supports application development with a business perspective in mind. Business logic is distributed in activities that each user can see and organize according to each project specification.
  Learn more…
- **Event-driven capabilities**. The Nirva Application Platform EVENT service allows applications to run in event-driven mode on a publish/subscribe model.
  Learn more…
- **Database**. Nirva Application Platform allows database communication using ODBC and JDBC via dedicated services.
  Learn more…
- **Mass storage**. Nirva Application Platform supplies a mass storage service that can be used to store or archive large amounts of data coming from containers or files.
  Learn more…

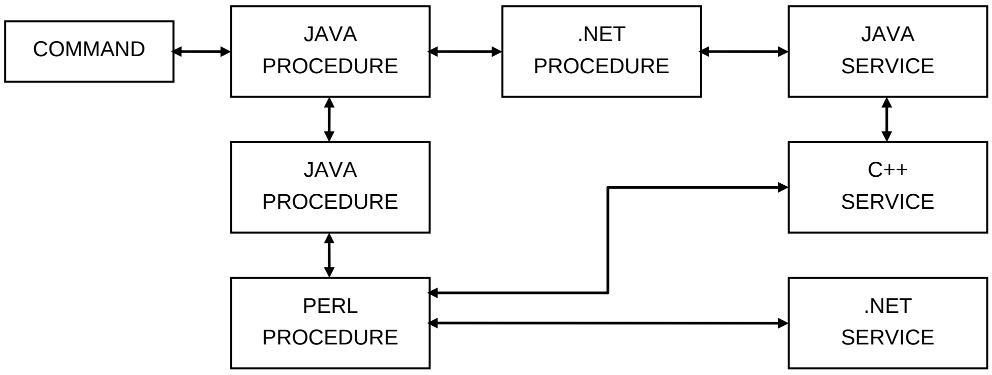
- **Registry**. A registry system interfaced with a Web editor allows storage of system configuration, application and service data.
  Learn more…
- **Configuration**. Product configuration is available through a Web interface or programmatically. All system parameters, but also applications and services can be configured this way.
  Learn more…
- **Monitoring**. Monitoring tools can be used to check system functions and to send automated alerts to an administrator in case of problems.
  Learn more…
- **Logs**. Nirva Application Platform supplies logs to track system or application processing. It is possible to add logs to any given application. Nirva Application Platform controls their size, their retention time and supplies search facilities.
  Learn more…
- **Multi-applications**. A single Nirva Application Platform instance can host several applications at the same time. Applications remain isolated from one another, have their own life cycle but can still communicate amongst themselves.
  Learn more…
- **Deployment**. Nirva Application Platform components (application, services, Web Services) can be packaged and deployed in few clicks from a simple Web browser.
  Learn more…
- **Multi-platform**. Nirva Application Platform is compatible with the following platforms: Windows, Linux, AIX, Solaris, HP-UX.
  Learn more…
- **Development tools**. Nirva Application Platform is programmed using standard development tools (Eclipse, Visual Studio, etc…). Nirva Application Platform also supplies tools to fine tune, debug and test.
  Learn more…

Multiple languages

Thanks to its structure, the product supports several programming languages, even within the same application. Supported languages are:

- Java
- .Net (all .Net supported languages)
- Perl
- C++

For simple procedures that do not need specific logic, it is also possible to use simple text files containing commands.



This unmatched flexibility allows for rapid and efficient integration of legacy programs. It also supports the use of specific technology only available in a particular language.

Java and .Net can cohabit in Nirva Application Platform.

Distributed Architecture

Nirva Application Platform is built as a distributed architecture to support application loads. An application can be deployed on one or several servers that can be dedicated to particular tasks.

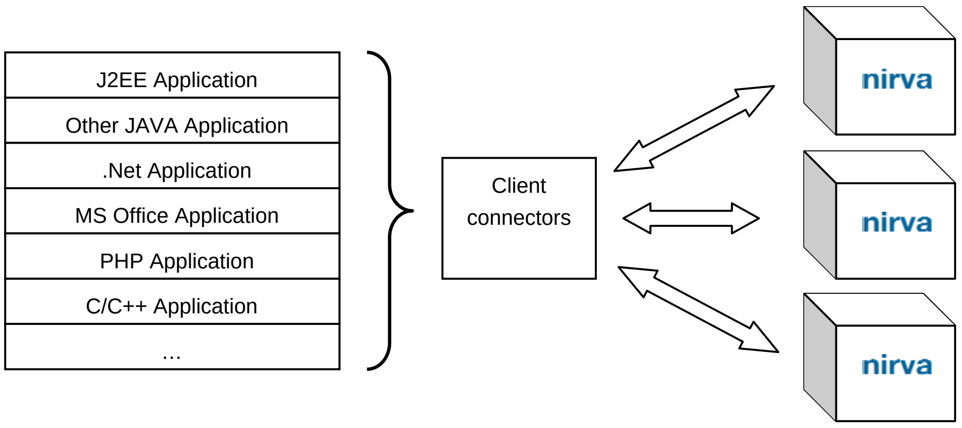
Servers can support load balancing and failover architectures.

Channels between the various servers allow fast and simple communication between them.

There are several configurations for load balancing and failover:

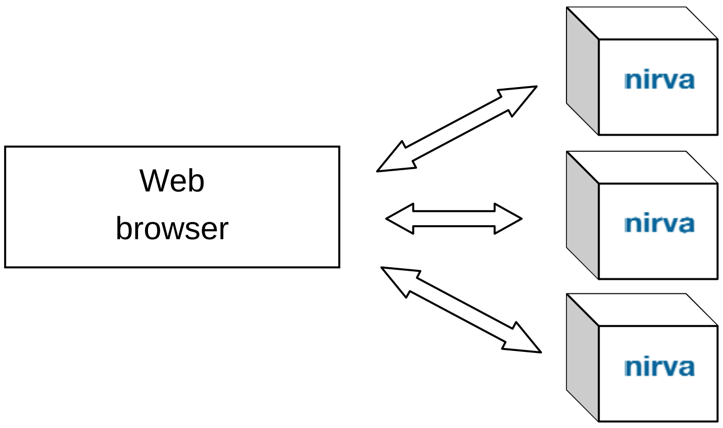
- Client connector load balancing
- Web front end load balancing
- Nirva Application Platform to Nirva Application Platform load balancing
- Nirva Application Platform to Nirva Application Platform with failover
- Message bus

Client connector load balancing

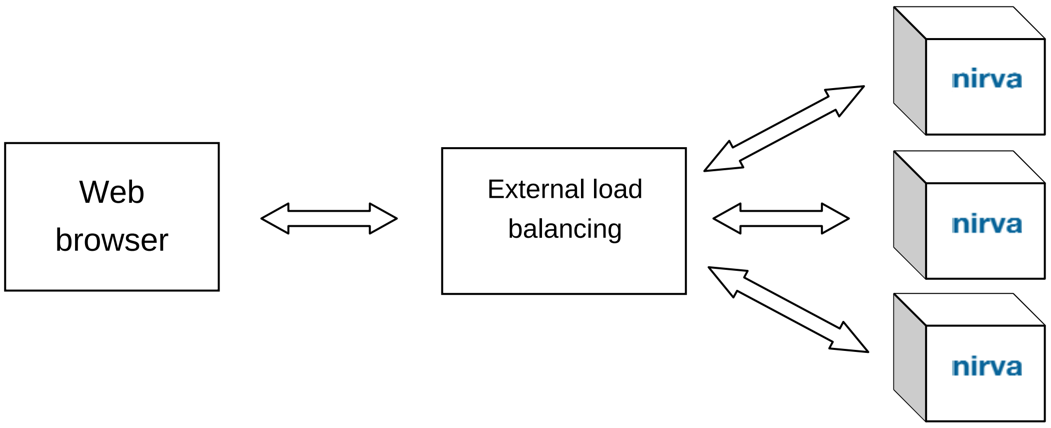


Web front end load balancing

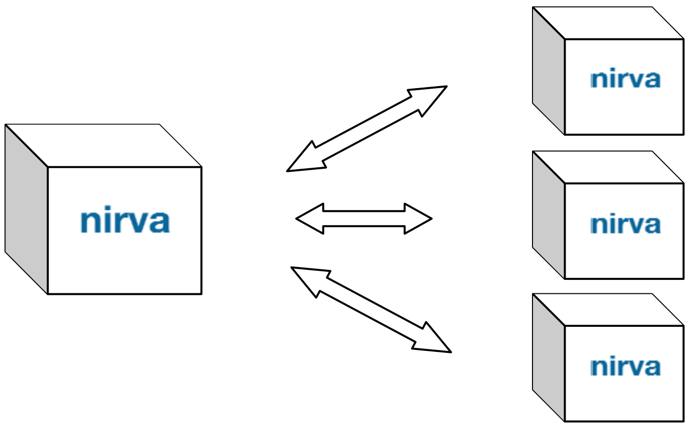
Nirva Application Platform uses HTTP redirection for Web load balancing:



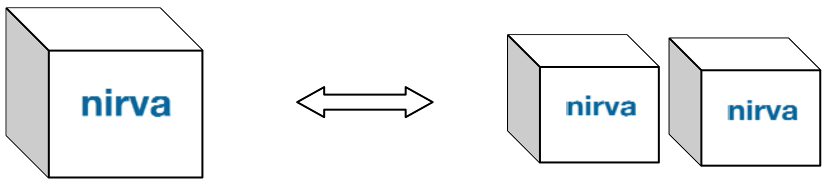
Load balancing can also be provided by dedicated external software or hardware solutions synchronized on the Nirva Application Platform session.



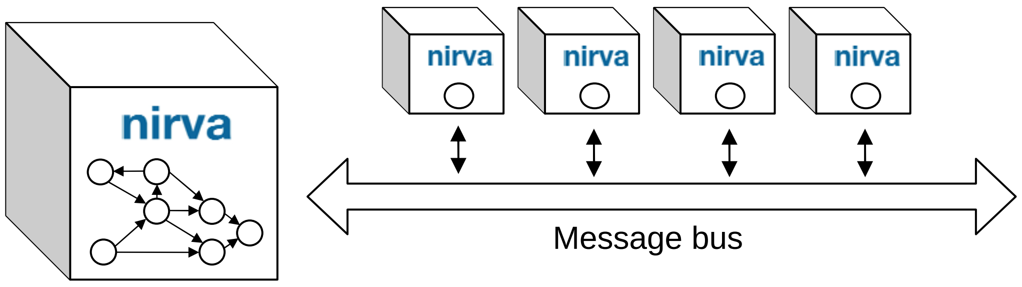
Nirva Application Platform to Nirva Application Platform load balancing



Nirva Application Platform to Nirva Application Platform with failover



Message bus



Standards

Nirva Application Platform is based on standards and proven components:

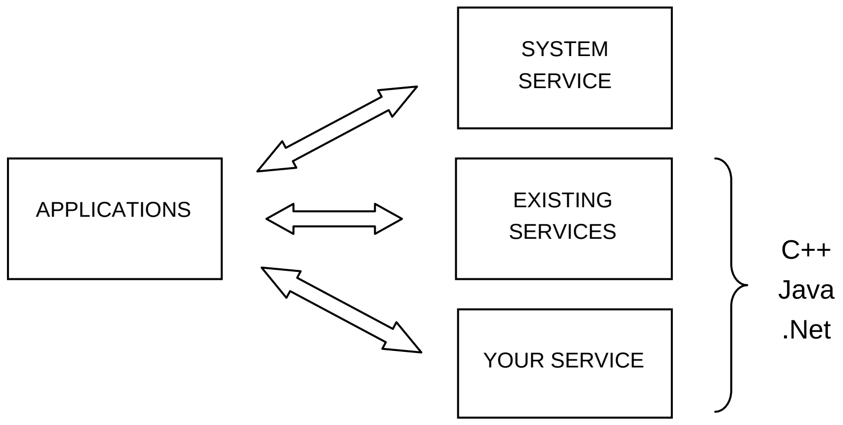
- HTTP/HTTPS
- XML
- XSLT
- SOAP
- WSDL
- Java
- .Net
- Perl

Extensible

The product functionality can easily be extended thanks to the service component.

The Nirva Application Platform service is a transverse component with its own life cycle and can be used to add new commands to the system. These commands can then be used by applications as any other command of the internal Nirva Application Platform service. A service can support either additional technology or a business process.

Services can be developed in Java, .Net or C++.

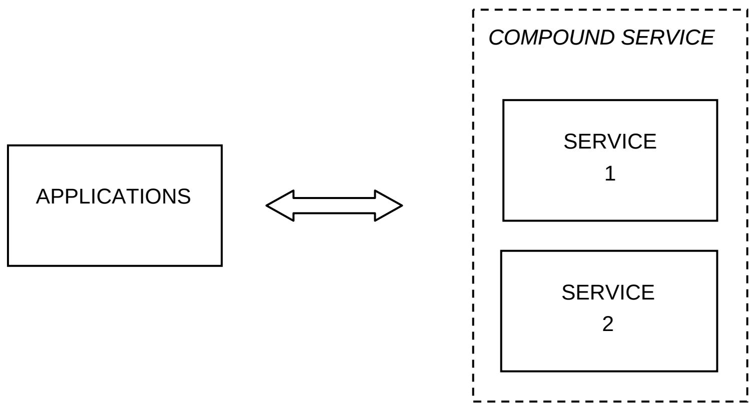


Nirva Application Platform supplies a license management tool that external partners can use to control their own services' licensing. An external service supplier can easily "sell" its added value and control its deployment.

Here are a few examples of Nirva Application Platform services:

- DATABASE (ODBC access to databases)
- JDBC (JDBC access to databases)
- WORKFLOW (orchestrating processes)
- STORAGE (mass storage)
- EVENT (event management)
- PDF (PDF file manipulation)
- Etc.

Services can call commands of other services. This feature can be used to create compound services. For example we can build an ARCHIVE service that uses the DATABASE and STORAGE services.

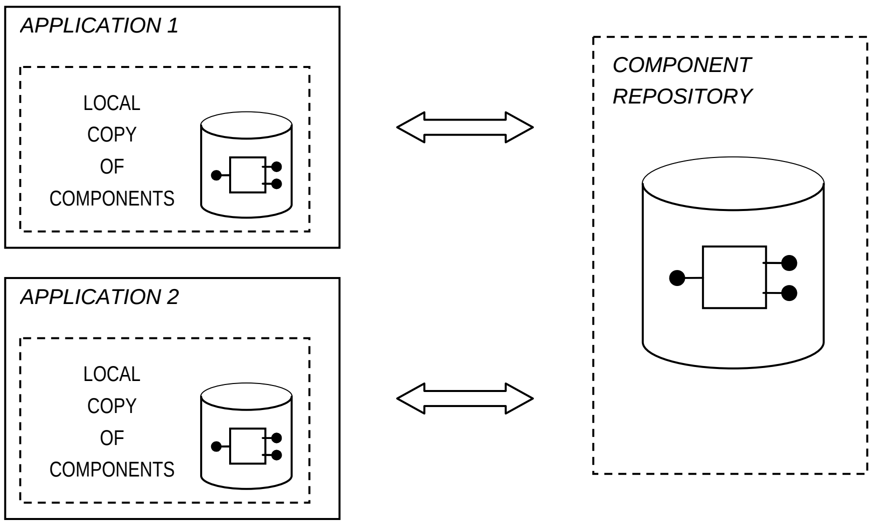


A developer can create a Nirva Application Platform service in few clicks and start programming it from an automatically generated skeleton code. Nirva Application Platform services are installed and configured from the main Nirva Application Platform Web configuration tool.

Component reusability

Components can be reused to share the common parts of applications. This boosts application agility and significantly reduces conception and maintenance costs. Reusability is possible by creating Nirva Application Platform services (transverse components with their own life cycle) or by using a specialized service (the NASC service or Nirva Application Platform Application Software Component).

The NASC service manages common components as versioned code used by applications. They can either be processing or presentation components. They are automatically loaded when an application starts.

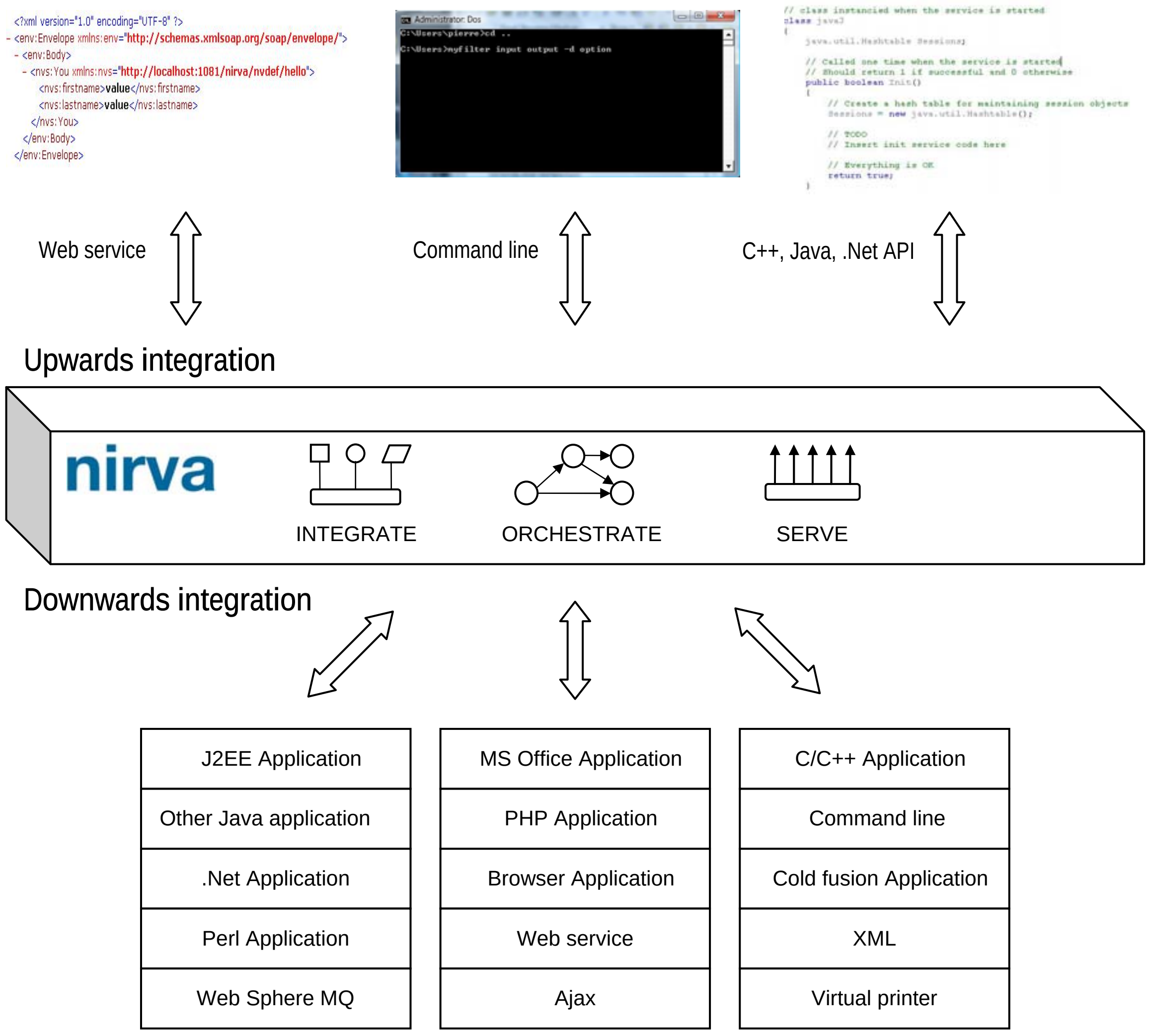


Upwards and downwards integration

Nirva Application Platform can integrate components, applications or external technology to build composite applications (upwards integration) but can also integrate these applications in other applications with its connectors (downwards integration).

Upwards integration can be achieved with:

- The creation of a service (integration via the API of the target application or technology).
- The connection to a Web Service or any other form of XML-based communication.
- A program launched from a command line.


```
<?xml version="1.0" encoding="UTF-8" ?>
- <env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
  - <env:Body>
    - <nvs:You xmlns:nvs="http://localhost:1081/nirva/nvdef/hello">
        <nvs:firstname>value</nvs:firstname>
        <nvs:lastname>value</nvs:lastname>
      </nvs:You>
    </env:Body>
  </env:Envelope>
```

```
C:\Users\pierre>cd ..

C:\Users>myfilter input output -d option
```

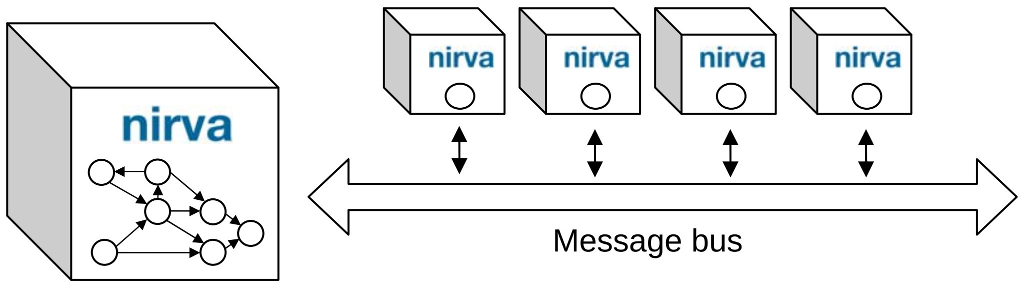
```
// class instancied when the service is started
class java3
{
    java.util.Hashtable Sessions;

    // Called one time when the service is started
    // Should return 1 if successful and 0 otherwise
    public boolean Init()
    {
        // Create a hash table for maintaining session objects
        Sessions = new java.util.Hashtable();

        // TODO
        // Insert init service code here

        // Everything is OK
        return true;
    }
```

Message bus

Nirva Application Platform contains a message bus to support process management in asynchronous mode.

In this mode, a central entity stores messages (activities) in queues where specialized agents can pick them up for processing.

Each agent processes the activities according to its capacity. This mode can be used to increase application scalability since agents can be easily added (or extra resources made available for existing agents) to boost performance.

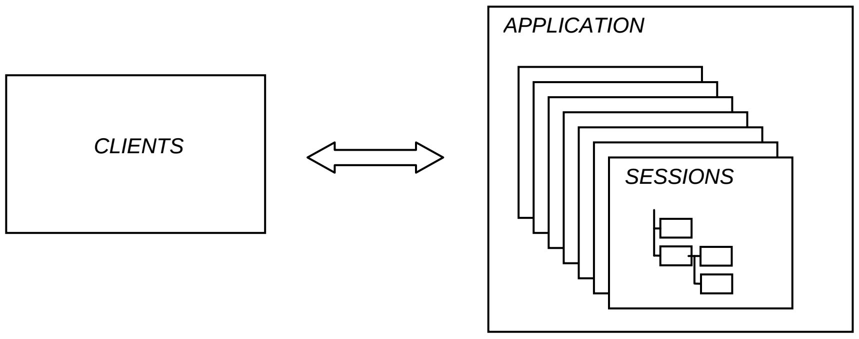


"Message bus" processing mode is supported by the Workflow and Event services.

Session management

Nirva Application Platform supports the concept of a session that represents a user's context. An application can only be connected through a session. There are different types of sessions:

- Client: user initiated session from a connector or a Web browser.
- Named: session shared between users. This session type can be used to create database connection pools.
- Scheduler: session initiated by the integrated scheduler.
- Listener: session waiting for an event.
- Thread: session asynchronously triggered in the background.
- Internal: session manually created by a dedicated command from a procedure or service.
- System: session created by the Nirva Application Platform engine itself.

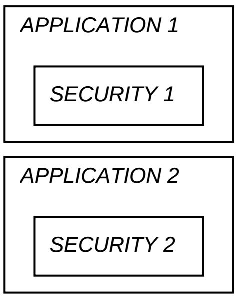


Sessions allow concurrent multiple users (a single server can handle hundreds of them) and the total isolation of data for each user. The user data is located in session containers.

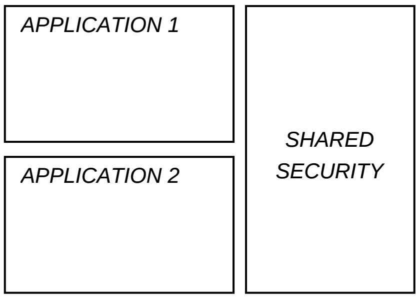
Security

Nirva Application Platform supplies security functions as a standard. The security model is based on permissions, roles and users. Security includes SSO (Single Sign On management) to allow automated authentication of users already connected on a particular domain.

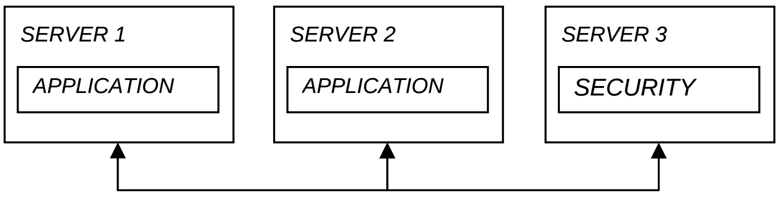
Security can be specific to an application, shared between several applications or shared between several servers.



Specific security



Shared security



Centralized security

The security model can be extended or entirely replaced by an external security system. For that purpose, a security service can be used to replace the standard security in Nirva Application Platform. Of note is a security service for LDAP.

Web Services

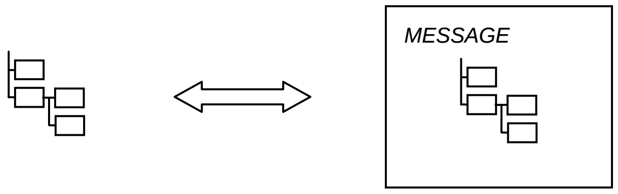
Nirva Application Platform is both a producer and consumer of Web Services.

Using Web Services is supported with the help of the integrated XSLT processor that transforms Nirva Application Platform input and output, and with the XML generator that converts container data to XML format.

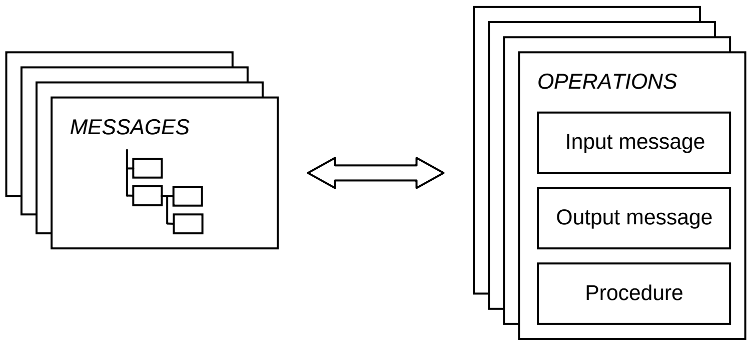
Web Service creation is extremely simple. A Web interface is used to define the structure of the Nirva Application Platform containers for input and output messages and to define a procedure to execute the relevant Web Service. Nirva Application Platform automatically generates the WSDL code describing the Web Service.

This occurs in 4 steps:

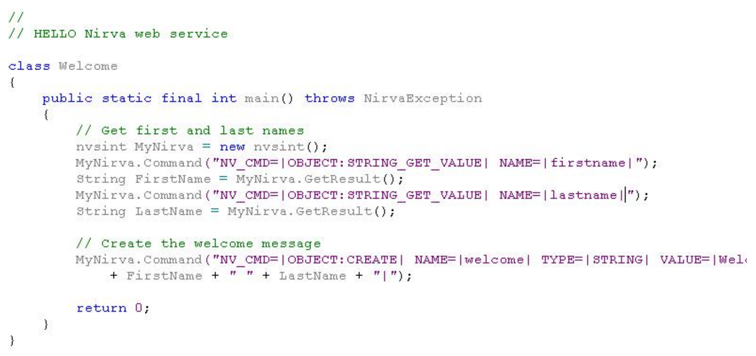
- Define containers for messages (Web interface).
- Associate messages with operations (Web interface).
- Create and code the procedure for each operation.
- Publish the Web Service (Nirva Application Platform automatically creates the WSDL).



1. Define containers for messages

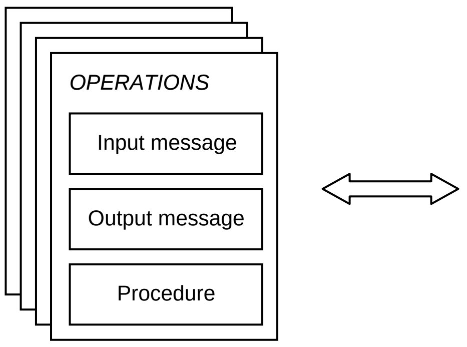


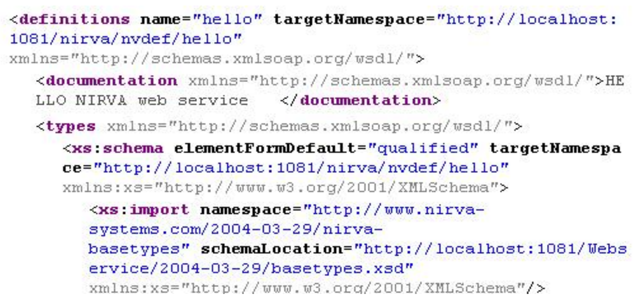
2. Associate messages with operations



```
//
// HELLO Nirva web service

class Welcome
{
    public static final int main() throws NirvaException
    {
        // Get first and last names
        nvsint MyNirva = new nvsint();
        MyNirva.Command("NV_CMD=|OBJECT:STRING_GET_VALUE| NAME=|firstname|");
        String FirstName = MyNirva.GetResult();
        MyNirva.Command("NV_CMD=|OBJECT:STRING_GET_VALUE| NAME=|lastname|");
        String LastName = MyNirva.GetResult();

        // Create the welcome message
        MyNirva.Command("NV_CMD=|OBJECT:CREATE| NAME=|welcome| TYPE=|STRING| VALUE=|Wel
            + FirstName + " " + LastName + "|");

        return 0;
    }
}
```

3. Code the procedure for each operation





```
<definitions name="hello" targetNamespace="http://localhost:
1081/nirva/nvdef/hello"
xmlns="http://schemas.xmlsoap.org/wsdl/">
   <documentation xmlns="http://schemas.xmlsoap.org/wsdl/">HE
   LLO NIRVA web service   </documentation>
   <types xmlns="http://schemas.xmlsoap.org/wsdl/">
      <xs:schema elementFormDefault="qualified" targetNamespa
      ce="http://localhost:1081/nirva/nvdef/hello"
      xmlns:xs="http://www.w3.org/2001/XMLSchema">
         <xs:import namespace="http://www.nirva-
         systems.com/2004-03-29/nirva-
         basetypes" schemaLocation="http://localhost:1081/Webs
         ervice/2004-03-29/basetypes.xsd"
         xmlns:xs="http://www.w3.org/2001/XMLSchema"/>
```

4. Publish the web service. Nirva automatically creates the WSDL.

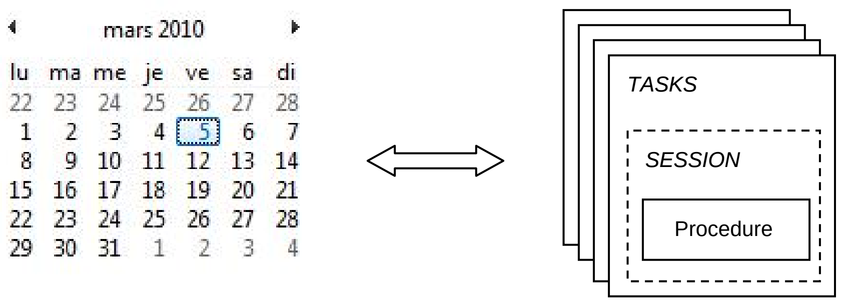
Nirva Application Platform security specifies which user can access a particular Web Service.

Applications, through Web Services, can communicate with the external world using recognized standards.

Scheduler

A scheduler is integrated in the core system and supports the planning of various tasks during a day, a week, a month or a year.

The tasks can be repeated indefinitely or a fixed number of time during a defined interval.

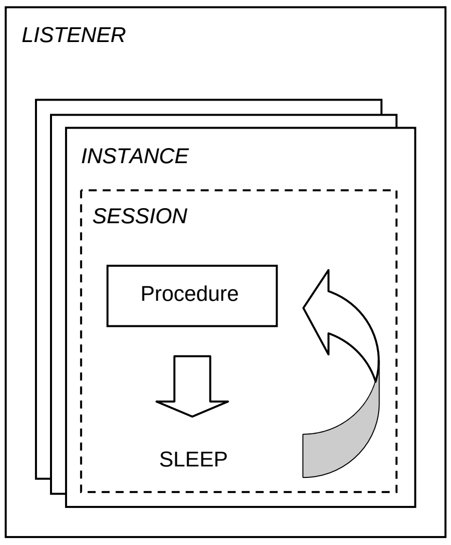


Using the scheduler reduces license costs as no external product is necessary.

Listeners

Listeners wait for events. They can be used for example to watch a directory for retrieving files to store them in the application to be processed.

Several instances of each listener can be executed in separate threads. A Listener runs a procedure in a loop with a defined sleep time between each occurrence.



Listeners are heavily used in distributed applications since they can be used to adjust additional application load. It is possible to define several instances of a particular listener and to distribute them on multiple machines.

They are often used for managing workflow activities in a message bus context.

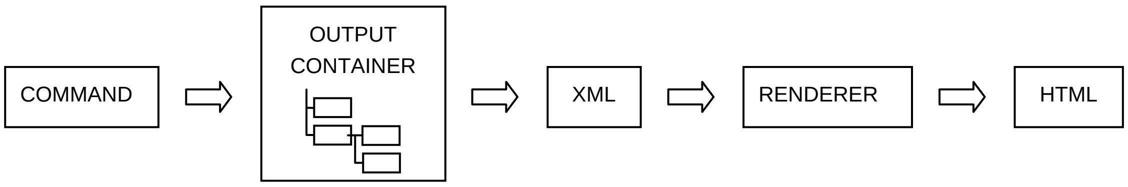
Presentation layer

Nirva Application Platform provides two ways for building Web presentation layers:

- HTML renderers
- Nidget framework

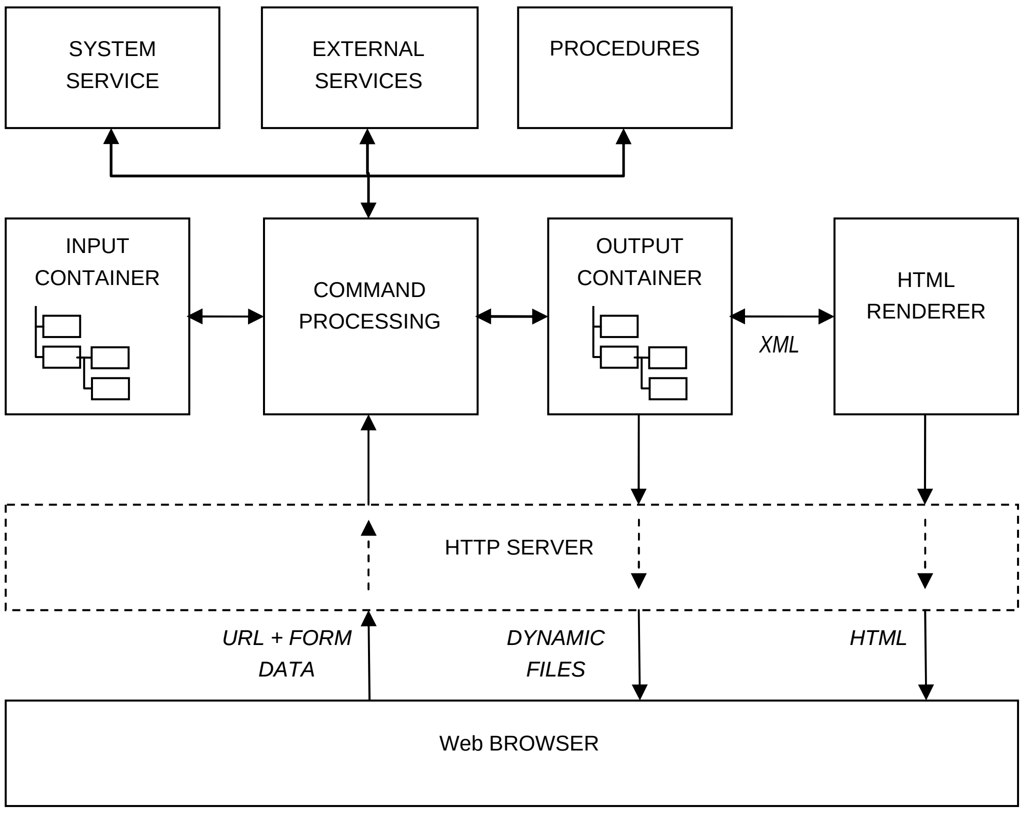
Html renderers

The presentation layer works with HTML renderers that transform the content of an output container to HTML flow.



Nirva Application Platform provides built-in renderers and delivers an interface for third parties to write their own renderers.

This is the typical request for a dynamic page. Static pages are directly handled by the HTTP server.



The Web browser sends a command to Nirva Application Platform as a GET or POST HTTP request with optional form data. A typical Nirva Application Platform URL command is the following:

```
http://myserver:1081/nv_app_myapp/NVS?command&NV_CMD=MYSERVICE:MYCLASS:MYCOMMAND&NV_PROC
=java:myproc&NV_XML_XSL=MYXSL&NV_SESSION_ID=1234567890
```

On receiving this request, Nirva Application Platform first connects the session identified by the NV_SESSION_ID parameter. If this parameter is not given, Nirva Application Platform opens a new session for the application “MYAPP” (other parameters authenticating the user must then be given).

The optional URL form data is transformed into session variables or file objects if there is a file upload and are reachable from the procedures and services.

As the NV_PROC parameter gives a name of a Java procedure (java:myproc), Nirva Application Platform executes this procedure. The NV_PROC parameter contains the name of one or several procedures that are executed before the command itself. Another parameter allows execution of other procedures after the command. A procedure contains itself some Nirva Application Platform commands and some calls to other procedures. Procedures are the place to build the business logic.

Then Nirva Application Platform executes the command given in parameter NV_CMD. This is the command named “MYCOMMAND” for the class “MYCLASS” of the service “MYSERVICE”. If the NV_CMD parameter is not given, Nirva Application Platform executes no command. This is generally the case when using Nirva Application Platform as a Web application server where the URLs just instruct Nirva Application Platform to execute procedures.

A command gets information from an input container and delivers its data to an output container. The container names are given as parameters of the command but there are some default values. In this exemple, both input and output containers point to the same default container. The procedures inherit the name of the input and output containers.

When the command has finished, Nirva Application Platform transforms the output container into XML and sends this XML stream to the renderer engine that generates the HTML code. In this example, Nirva Application Platform uses the default renderer which is XSLT. The NV_XML_XSL parameter of the URL gives the name of the XSLT style sheet to use.

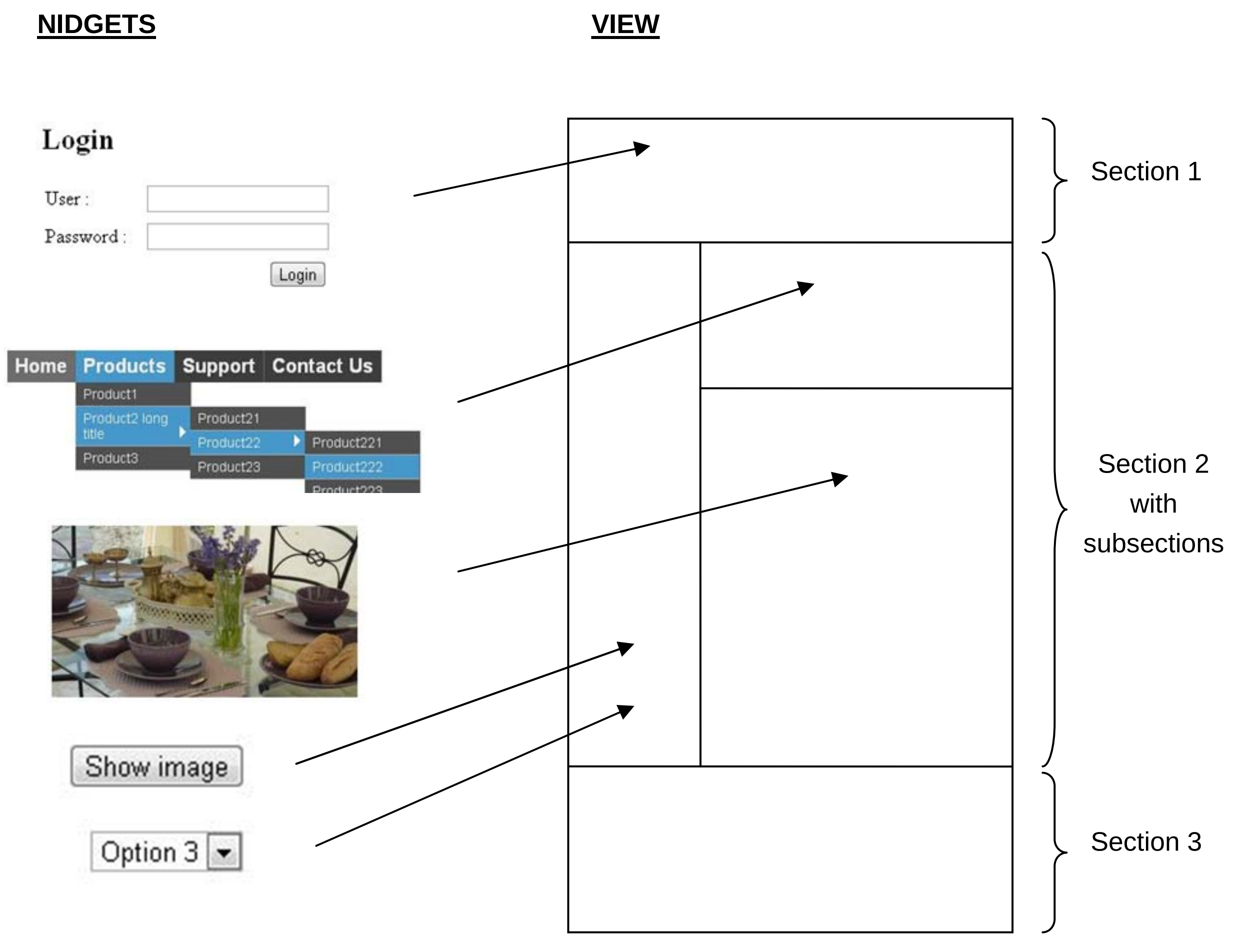
When using the client connectors from external applications, the command cycle is the same except that there is no HTML output. The output container data is directly accessed by dedicated commands.

This technology presents the key advantage of separating the processing part from the presentation part. Development and maintenance are easier.

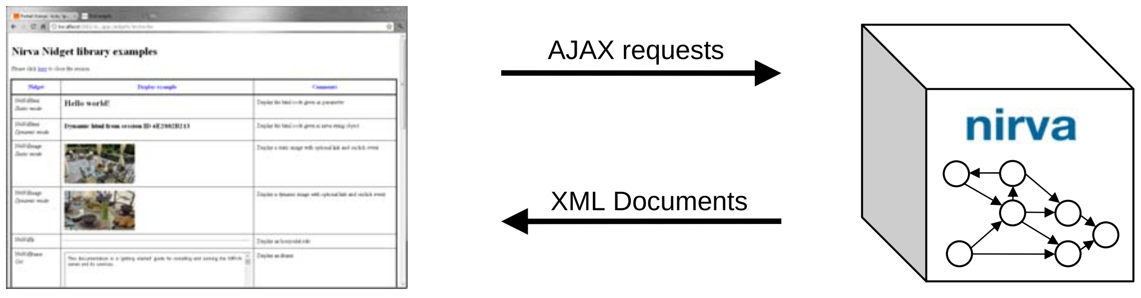
Nidget framework

The Nirva Application Platform nidget framework is based on JavaScript and Ajax technologies. It is a set of functionality allowing fast creation of Nirva Application Platform Web applications. “Nidgets” is the contraction of “Nirva Application Platform” and “Widgets”. A Nidget is a standalone and customizable Web component written in JavaScript. Nidgets are organized in libraries.

The framework defines the concept of view. A view is a Web page divided in sections and subsections like nested tables in an html page. Each cell of the view can be populated with one or several nidgets.



The dynamic data exchange between Nirva Application Platform and the browser is Ajax based. A view defines one or several XML documents generated by Nirva Application Platform. Each nidget can be associated to one or several documents. If a document changes, an event is sent to the attached nidgets allowing them to change their display.



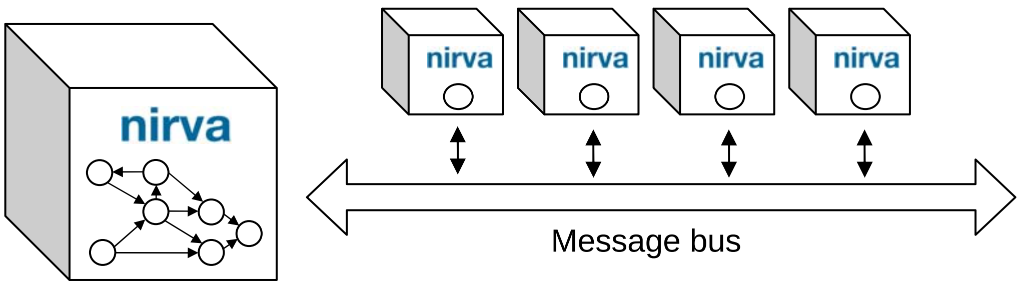
The Html code of the views is kept in a memory cache for being displayed in a very fast way. The code of the view is very short because it only contains the layout and names of the nidgets used in the view. All the real code is inside the nidgets themselves in Javascript files directly loaded and cached by the browser.

Once compiled and published, the framework code does not change, so the display of Web pages is fast.

Workflow

The Workflow service supports application development with a business perspective in mind. Business logic is distributed in activities that each user can see and organize according to each project's specification.

The service only manages the orchestration of activities. The process itself is handled by specialized agents connected to the message bus. These agents can be Nirva Application Platform servers or any other application connected to Nirva Application Platform through one of its available client connectors.



Key features include:

- Creation and deployment of workflows.
- Business logic defined in the activity flow.
- Synchronous and asynchronous activities.
- Process number (workflow cases) unlimited.
- No need for an external database.
- Audit functions supported.
- Business data correlation to identify processes.
- Data storage available to each process.
- Web configuration and management.

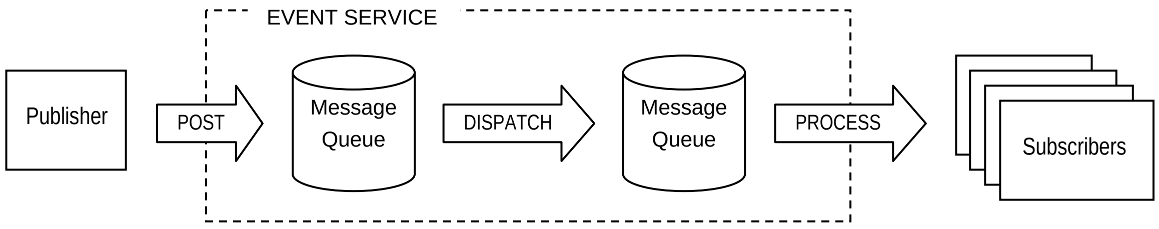
Event-driven capabilities

The EVENT service is a Nirva Application Platform external service which provides Nirva Application Platform with a message tracking system allowing Event Driven Architectures to be built. It keeps track of messages (events) sent to a channel and the subscribers who are to receive the events. The service allows multiple occurrences of the same messages to be received allowing for escalation mechanisms. It gives the subscribers the possibility of inhibiting further occurrences of a given message.

The EVENT service allows:

- Having multiple channels open to publishing events and event subscriptions
- Processing multiple occurrences of the same messages allowing escalation mechanisms to be configured in case of multiple publications of the same messages (in particular in the case of events being alerts to be processed).

- Management of multiple subscribers per channel and generation of specific actions for each subscriber.
- Inhibiting occurrences of a message on a per-subscriber basis.
- Different possibilities for desynchronizing the posting of an event and its processing.



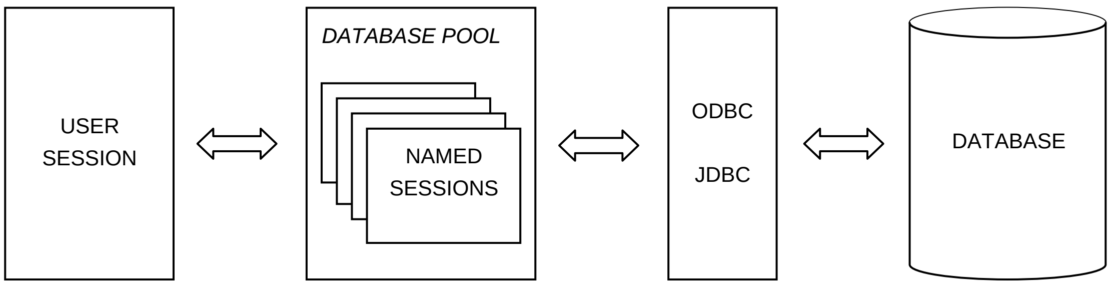
Database

Nirva Application Platform allows database communication with ODBC and JDBC using dedicated services. Other methods are also possible via alternate Java or .Net components.

Database table data is mapped to Nirva Application Platform objects allowing easy access.

Nirva Application Platform provides some features for creating pools of database connections that can be shared by the users.

The SQLite file database engine is directly embedded in the Nirva Application Platform kernel.



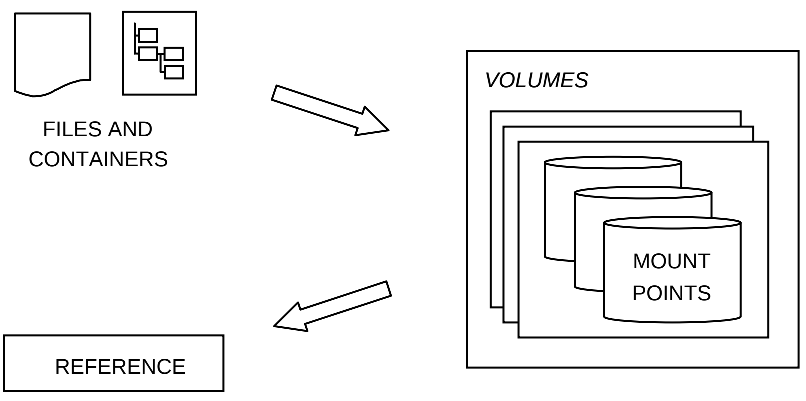
Mass storage

Nirva Application Platform supplies a mass storage service that can be used to store large amounts of data coming from containers or files.

When storing information, the storage volume returns an identifier that can be recorded with the meta-data in a database and used to retrieve the information.

In the case of storing a container, the service allows retrieval of all or only some objects in the container.

This service can manage logical volumes with one or several physical mount points and supports replication between these mount points.



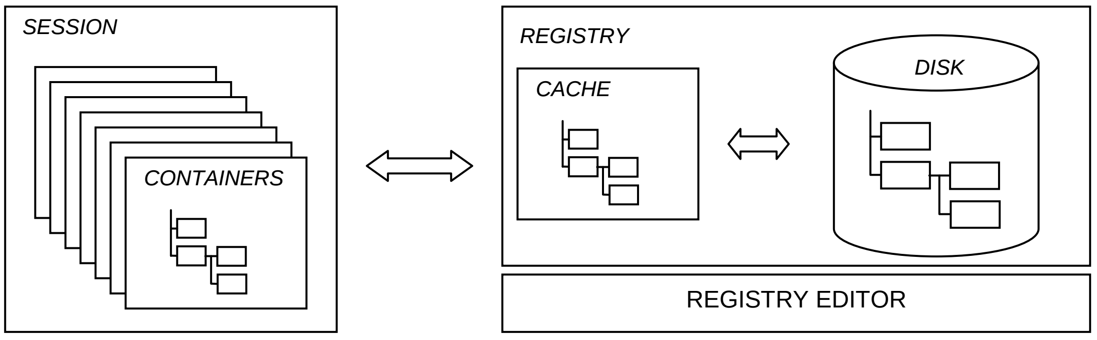
Volumes can be grouped to boost system extensibility and performance.

The mass storage service is usually implemented in failover mode to guarantee business continuity in case of failure.

Registry

A registry system interfaced with a Web editor allows storage of system configuration, application and service data. The registry can also be used to store business data on smaller systems.

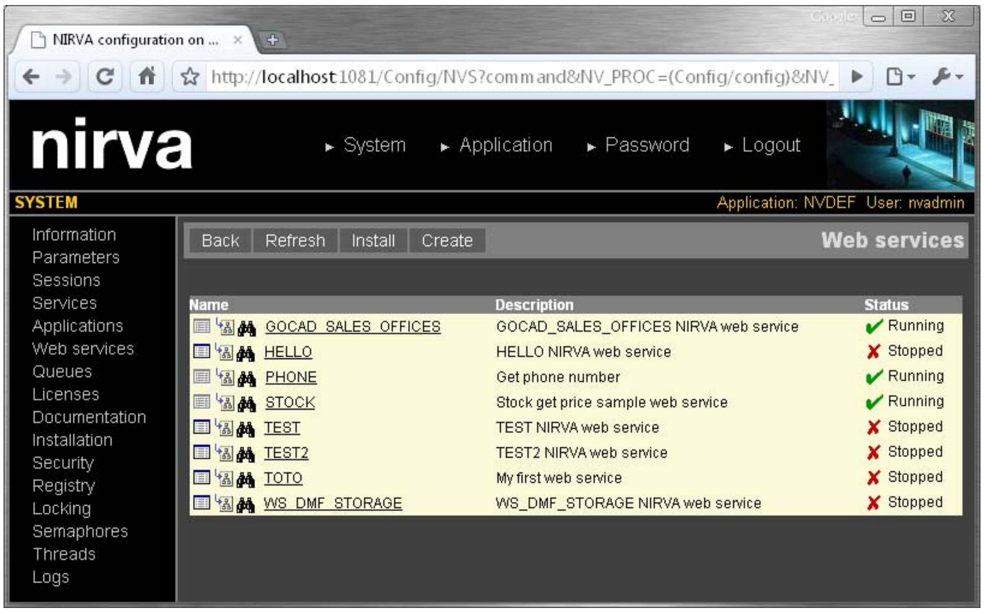
A registry is a persistent container. Nirva provides a registry cache for fast access.



Using the registry saves development time as it becomes unnecessary to create application configuration visual tools.

Configuration

Product configuration is available through a Web interface. All system parameters, but also applications and services can be configured this way.



Name	Description	Status
GOCAD_SALES_OFFICES	GOCAD_SALES_OFFICES NIRVA web service	✓ Running
HELLO	HELLO NIRVA web service	✗ Stopped
PHONE	Get phone number	✓ Running
STOCK	Stock get price sample web service	✓ Running
TEST	TEST NIRVA web service	✗ Stopped
TEST2	TEST2 NIRVA web service	✗ Stopped
TOTO	My first web service	✗ Stopped
WS_DMF_STORAGE	WS_DMF_STORAGE NIRVA web service	✗ Stopped

Configuration can also be automated by using the API from the client connectors. This simplifies processing.

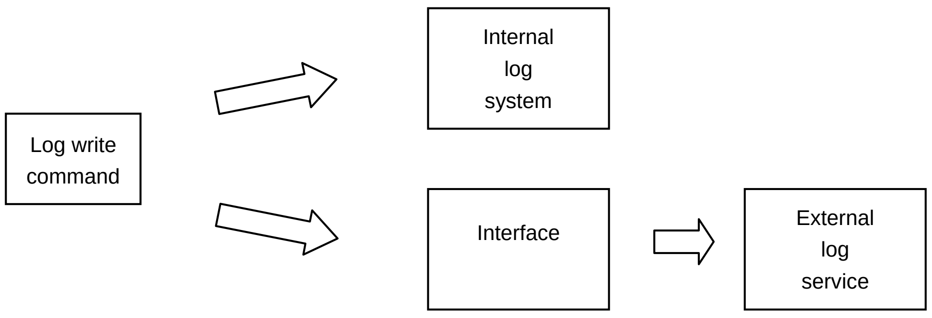
Monitoring

Web based monitoring tools can be used to check system functions.

Logs

Nirva Application Platform supplies logs to track system, service or application processing. It is possible to add logs to any given application. Nirva Application Platform controls their size and retention time, and supplies search facilities.

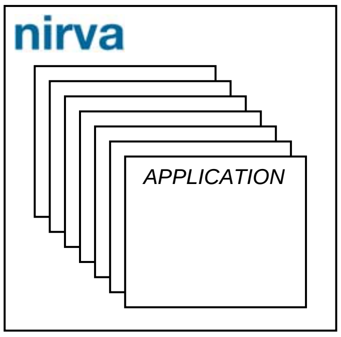
The internal log system can be replaced by a dedicated log service. Nirva Application Platform provides an interface for creating log services.



This integrated functionality obviously does not need to be developed for each new application. This significantly reduces development time and overall costs.

Multi-application

A single Nirva Application Platform instance can host several applications at the same time. Applications remain isolated from one another. Each has its own life cycle but they can still communicate amongst themselves.



This functionality can be used to separate certain parts of a particular project in order to better support their evolution. Alternatively, this can also be used to host several instances of the same application for different users (multi tenancy).

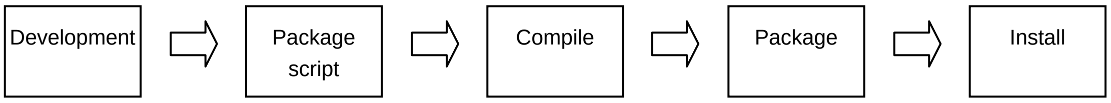
Deployment

Nirva Application Platform components (application, services, Web Services) can be packaged and deployed in few clicks from a simple Web browser. It is also possible to package only part of components (ex patches).

Nirva Application Platform provides a simple script file for defining component packages. The default package script is generally enough for packaging components but it can be easily modified to suit requirements.

Once the package script has been created, it can be compiled from a command line, an API or a Web interface to produce a package file.

The package file can then be deployed on the target system.



Multi platform

Nirva Application Platform is compatible with the following platforms:

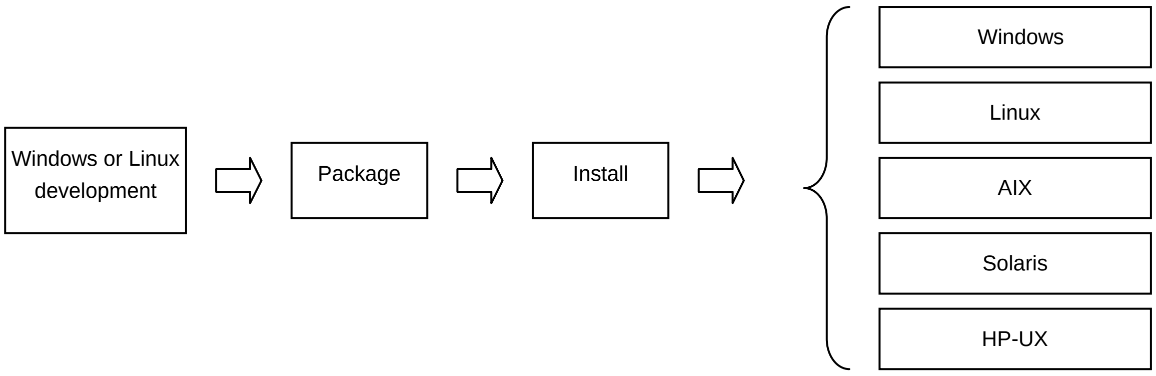
- Windows
- AIX
- Linux
- Solaris

- HP-UX (RISC and Itanium)

An application can run on any platform (unless it integrates platform specific components, e.g. procedures or services in .Net).

The choice of platforms helps in adapting the product to the customer's strategic choice. This is a key driver for Nirva Application Platform; it always strives to adapt itself to the existing technology as opposed to imposing its own choices.

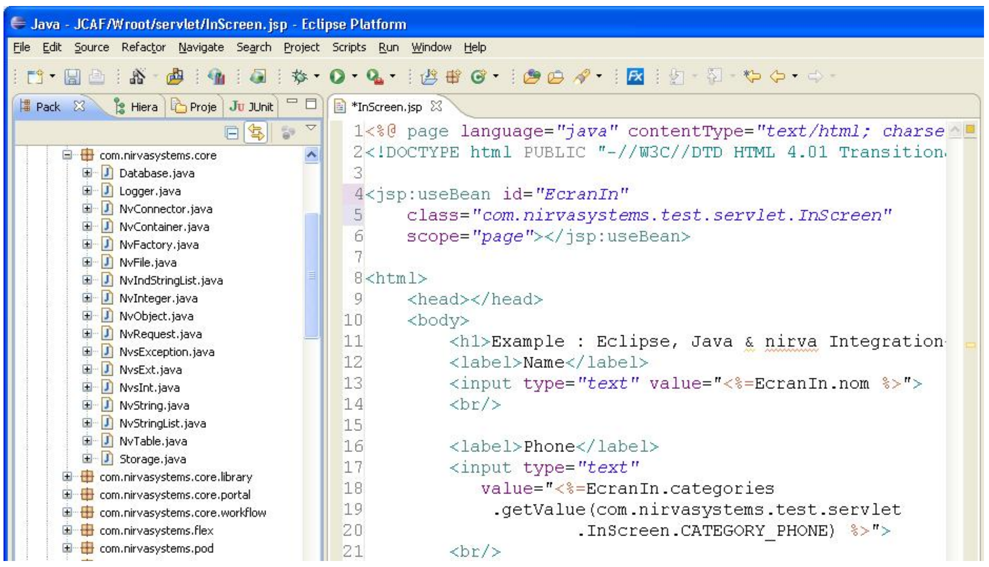
The possibility of running Nirva Application Platform applications in total independence of the platform guarantees minimal costs. When production imposes expensive environments (e.g. AIX, HP-UX), it remains possible to develop on cheaper platforms.



Development tools

Nirva Application Platform is relatively independent of the programming language and therefore of the development tools.

Development tools are related to the language used. Generally speaking, Eclipse is used for Java, Perl and C++ development. Visual Studio is most commonly used for .Net and C# Windows development.



```
<%@ page language="java" contentType="text/html; charse
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transition

<jsp:useBean id="EcranIn"
    class="com.nirvasystems.test.servlet.InScreen"
    scope="page"></jsp:useBean>

<html>
    <head></head>
    <body>
        <h1>Example : Eclipse, Java & nirva Integration
        <label>Name</label>
        <input type="text" value="<%=EcranIn.nom %>">
        <br/>

        <label>Phone</label>
        <input type="text"
           value="<%=EcranIn.categories
            .getValue(com.nirvasystems.test.servlet
                    .InScreen.CATEGORY_PHONE) %>">
        <br/>
```

Developer's guide documentation is supplied.

Nirva Application Platform also supplies some tools to fine-tune, debug and test an application to speed up development and maintenance time.