Open-Xchange™ Whitepaper

Open-Xchange Server 6
Architecture Overview

v1.3

Author: Stephan Martin
Editors: Dave Goldberg

Contents

1. Introduction

Open-Xchange Server 6 is the most advanced Linux based Groupware solution f.

Open-Xchange Server 6 is superior in the following areas:

- Open-Xchange Server 6 offers most comprehensive Groupware functionality based on modern user interface technologies and includes email, calendaring, contact management, intelligent document sharing, "smart linking" and much more.
- The architecture is designed for deployment in the hosted market.
- The architecture is designed for enterprises market

This document focuses on the later topic and describes the design goals of Open-Xchange Server 6 with regard to deployment in large scale, integrated, hosted- and enterprise environments.

2. Design Goals

Open-Xchange Server 6 was designed especially to fulfill the following requirements each of which is crucial for deployment in integrated hosting and enterprise infrastructures:

1. Scalability to millions of users
2. Multi Tenant / Multi Domain capabilities to serve hundreds of thousands small customers with one large data center
3. Integration into existing hosting infrastructure and services to leverage existing services and to integrate in the hosting company's business processes
4. Integration into enterprise environments
5. Offer the necessary services and interfaces to integrate into an existing data center for automated administration, deployment and monitoring with 24/7 uptime 365 days a year

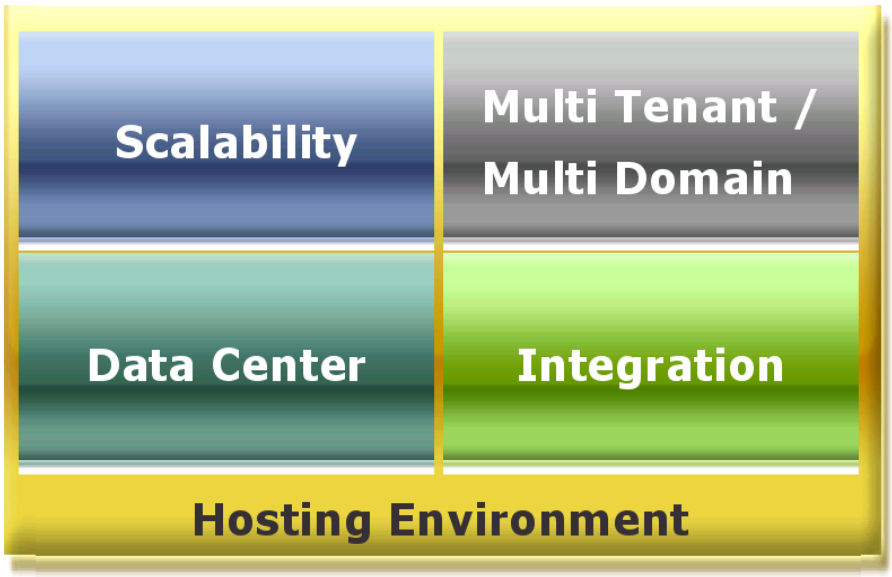

Figure 1: Open-Xchange - Design Goals

2.1. Scalability

Ultimately everything described in this document relates to scalability. Scalability has many different faces including serving hundreds of thousands of concurrent

users, efficiency of the overall deployment, reuse of existing services, resources needed for administration, and much more.

Open-Xchange Server 6 is designed to serve several hundreds of thousands of customers each with many individual user accounts. In very large environments, this leads to millions of users in a single installation.

To scale to these numbers, Open-Xchange Server 6 makes use of two modern scalability concepts, "Scale Up" and "Scale Out", both of which are described in more detail later.

2.2. Multi Tenant / Multi Domain

To meet the scalability requirements mentioned above it is necessary to make efficient use of the available hardware resources.

To meet high volume scalability requirements posed by the SaaS model and enterprise environments, one might be inclined to place each customer on their own dedicated server. However, this would lead to a huge waste of resources as many of the processes in each dedicated server would do the the same work as other processes in other servers. In general, dedicated servers and virtualization is not an efficient scaling technique for many small customers as is encountered in the SaaS model.

What is needed to solve this problem is the called Multi Tenant / Multi Domain capability. This allows many customers to access exactly the same service without seeing the existence of other customers in the same process. Every customers experience is as if they were the only customer using this service. This allows for the most efficient usage of hardware resources as the application takes care of the separation between the customers environments. Each end user can only see the users, objects and resources from within their own customer environment. Such an virtual environment is called a **context** throughout this document.

2.3. Integration

Another crucial topic with regard to scalability is the ability to leverage existing services. Although not a technical requirement of the software, the ability to re-use existing services reduces project scope, improves business processes and increases infrastructure efficiency. Any IT person who has implemented large scale environments knows that it is not efficient to reinvent or replace existing systems which already perform their task well and are designed specifically for the environment they run in.

The operation and deployment of Open-Xchange Server 6 is based on the ability to leverage existing services and processes in the background.

These services can be thought of in two distinct areas: customer facing services, like Email and business process related services like HR-Software. It is important for Open-Xchange to work smoothly with existing services rather then require them to be replaced.

Open-Xchange focuses on its core competency which is providing large scale groupware applications. All other services necessary to run Open-Xchange Server 6 can be integrated smoothly from the hosting or enterprise environment through the many available open and documented interfaces provided by Open-Xchange. There is no need to perform costly replacements of existing systems to utilize the groupware functions of Open-Xchange Server 6.

2.4. Data Center

Another important part of scalability is the ability to operate all the services in a very efficient manner allowing 24/7 operation with as small a staff as possible.

Open-Xchange Server 6 offers everything a modern enterprise solution needs to be integrated into existing administration, deployment and monitoring frameworks.

3. Architecture Overview

This chapter gives a more detailed view of the Open-Xchange Server 6 architecture. It dives deeper into the different Open-Xchange Server components and dependencies and how they work together

The following diagram shows the components in a typical Open-Xchange Server environment.

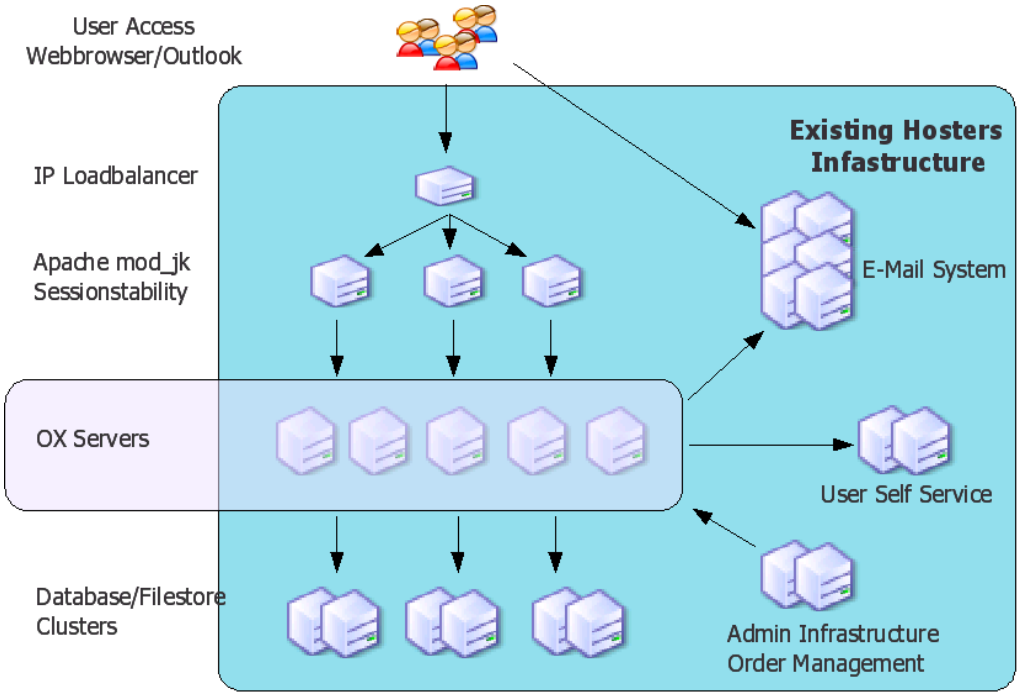

Figure 2: Open-Xchange - Simplified Architecture Overview

Open-Xchange includes the user front ends (AJAX based web front and rich client connectors) and all application logic. There are other additional services which are necessary for the operation of an Open-Xchange Server. Therefor an integrated approach is used to leverage a number of Linux services and frameworks. A running Open-Xchange environmental therefore includes both Open-Xchange components and components delivered from other 3rd party vendors.

Tracing how a user request is handled helps to understand all the components and how they interact together:

1. The user request is generated through a front end application.

- This is either an Internet browser running the Open-Xchange AJAX Graphical User Interface (GUI)

  or

- Microsoft Outlook© with the Open-Xchange OXtender for Microsoft Outlook© Plug-In.

2. The request is sent over the Internet (via HTTP or HTTPS) to an `apache` server. The `apache` server handles all Internet communication including request information, encryption, transmission errors and other low level processing details.
   See: `http://www.apache.org/`

3. Within `apache`, a module called `mod_proxy_ajp` is loaded and used to forward the request to the Open-Xchange application via the AJPv13 protocol. The primary task of `mod_proxy_ajp` is to ensure session stability when a cluster of Open-Xchange Server 6 is used. `mod_proxy_ajp` is part of the popular Jakarta project.

   See: http://tomcat.apache.org/connectors-doc/ajp/ajpv13a.html
   and: http://tomcat.apache.org/connectors-doc/

4. The core of the system is the Open-Xchange application server itself. All requests are handled, interpreted and fulfilled from this process running on a Linux server. The Open-Xchange Server 6 is written in JAVA and is running as an independent service. It is not embedded in a J2EE application server an thus avoids the overhead associated with those types of applications.

5. All data which is stored (appointments, contacts, documents, user configuration, authentication data, Emails, ...) is stored in back-end services which are designed specifically to store this type of data.
   In the Open-Xchange Server 6 there are three types of stored data and corresponding services:

   - All user data such as appointments or contacts is stored in a SQL database along with information about user authentication and company data.
     Open-Xchange Server 6 is based on the most popular open source database, MySQL. However, other SQL databases can be adapted for data storage.
     See: http://www.mysql.com/

- All documents are stored directly in the file system to avoid abusing the database systems to store large amounts of binary data. Only the meta data related to documents is stored in the database.
- All Emails are stored, read, and sent through standard Email services based on the standard Internet protocols IMAP and SMTP. Any IMAP and SMTP capable Email service can be integrated.

3.1. OSGi-Support

The framework implements a dynamic component model. Applications or components can be installed remotely, started, stopped, updated, and un-installed without requiring a reboot. This gives development and the hoster a more flexible working behavior. In addition, the Open-Xchange platform can easily be enhanced on top of the OSGi standard with own modules and features. There are several build-in modules which can be replaced with individual methods or even platform specific features:

- Authentication module
- Config-Jump
- Individual Servlets
- Mail Filter
- Spam Handler
- Mail Abstraction Layer

4. Architecture Details

This chapter focuses in detail on the topics mentioned above.

Each topic is discussed in detail below with the goal of achieving maximum scalability by efficient operation, not solely focusing on performance related topics.

4.1. Scalability

Scalability in a purely technical domain is related directly to the term "performance".
Thus we need to define "performance":

Performance in an Open-Xchange environment is measured by two important statistics:

1. The number of concurrent user session which are served by one complete clustered installation or by one single server
2. The number of user requests which are answered by one complete clustered installation or by one single server in a given time frame

Understanding how performance is maximized within Open-Xchange Server 6 one begins to understand how the server scales to handle increased load.

4.1.1. Deployment Flexibility

Initially a systems setup is designed and sized based on assumptions. In a production environment it is common that the original assumptions were incorrect or become invalid with time.

This problem is common with undersizing as well as oversizing. For example, the server load could end up much lower than expected due to customers moving away from the service over the time. In this case the setup may waste expensive hardware resources. On the other hand, it is possible that the load becomes higher than expected due to a large percentage of power users. In that case it is necessary to lower the load on the relevant subsystems to keep those machines responsive for all users.

Both cases outline the fact that flexibility with regard to deployment is an important part of any architecture.

Open-Xchange Server 6 can adapt the deployment to the real world situation in a flexible and transparent manner whenever it is called for. At any time it is possible to improve the configuration based on the real world operation.

4.1.2. Scalability concepts definition

As described above the Open-Xchange Server 6 is designed to allow maximum efficiency through both flavors of scalability to increase the overall performance:

- "Scale Up", also called "Vertical Scalability"
- "Scale Out", also called "Horizontal Scalability"

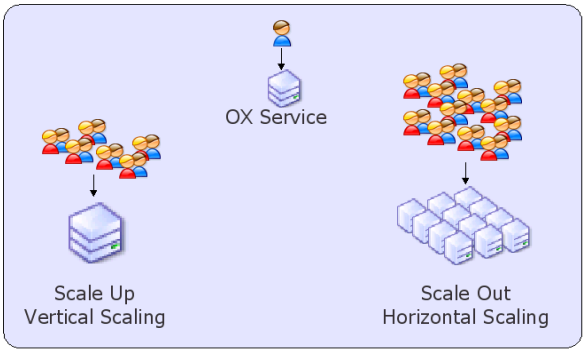

Figure 3: Open-Xchange - Scalability Concepts

"Scale Up" or "Vertical Scaling" means, adding more hardware resources (CPU, RAM, I/O, ...) to one machine to deliver more performance.

Obviously there are limits for this kind of scalability as it is not possible to achieve unlimited scalability through using better hardware. This is primarily because CPU and RAM are not the only factors which limit scalability. Additionally, continually buying bigger and better hardware is quite expensive and does not take advantage of the existing investment.

When this point is reached the other concept needs to be applied: "Scale Out" or "Horizontal Scaling".

"Scale Out" or "Horizontal Scaling" means that the performance of the overall system is increased by adding more machines to the system (clustering, service separation). With this concept it is possible to theoretically achieve "unlimited" scalability.

For a truly robust, scalable solution both concepts need to be properly supported. If the application is not designed properly, it does no good to add more CPUs or RAM into the server. Likewise a poorly designed architecture will not benefit from adding more machines to increase overall performance.

Open-Xchange Server 6 makes use of both concepts to allow maximum scalability through a superior service architecture.

4.1.3. "Scale Up" or "Vertical Scaling"

To allow maximum scalability with a single machine it is important to design an application to utilize the existing hardware resources as efficiently as possible and not to waste any resources.

There are several concepts implemented in Open-Xchange Server 6 which lead to efficient scalability on one single machine.

All of the following contribute to help scalability and each has been carefully adopted by Open-Xchange Server 6:

- Reduce the computing power to fulfill a special task
- Reduce the memory usage of the application
- Reduce all requests between components (internal components, external services)
- Reduce unnecessary tasks all together

The above mechanisms are accomplished via the following technologies:

- **AJAX Frontend**
  Unlike the classical web applications, the user interface is not generated by the server. AJAX technology allows rendering the complete front end, written in Java-Script, directly within the users browser. Only data objects are exchanged between the application running in the browser and the server. The data objects are transferred using a format called Java-Script Object Notation (JSON) to reduce the parsing effort in the browser to a bare minimum.
  See also: `http://www.json.org/`
  This new concept reduces the load on the server dramatically when compared to classical web applications where all the rendering work and the generation of the HTML code is done on the server.
- **Database Connection Pooling**
  Another potential bottleneck and potential waste of resources is the handling of database connections. Large numbers of user requests require large numbers of database requests to the back end database server.
  Open-Xchange Server 6 makes use of a very efficient database connection pooling mechanism to allow efficient reuse of existing connections to the database server and to reduce the overhead of handling these connections in the JDBC driver (CPU, RAM).

- **Efficient Caching**
  Another scheme of optimizing the use of data is the caching of frequently used data structures from the database.
  Open-Xchange Server 6 leverages the database's internal caching mechanisms through "caching ready" Open-Xchange JAVA objects. This translates to a tremendous reduction in the load on the server because often accessed data structures, like permission sets, are cached internally as Open-Xchange objects. This reduces not only the database accesses, but also the interpretation of the raw database data into Open-Xchange objects. The caching is based on the Java Caching System (JCS) from the Jakarta project. To allow the caching to work efficient in a clustered environment Open-Xchange uses cache invalidation technology implemented in JCS.
  See also: http://jakarta.apache.org/jcs/
- **Trigger/Push Mechanisms**
  Asynchronous connected front ends like the OXtender for Microsoft Outlook© need to keep their data in sync with the data of other users. To achieve this it is necessary to reload the data from the server regularly. Having many Outlook© clients polling in regular time frames can create a very high load on the server even if no data has changed.
  To avoid this regular polling an event trigger mechanism is introduced which triggers a push from the client to the server only when an object is modified on the server.
  This mechanism reduces server and network load tremendously as the server only uses computing power if there is actually some data to synchronize.
- **IMAP Caching**
  IMAP operations on the IMAP back end server are typical bottlenecks in webmail implementations. This is reduced to a minimum by Open-Xchange through caching the most important data from the IMAP mailbox. For every mailbox there is only one single request to read the content. This information is cached in Open-Xchange. The user will always see his list of emails without the need to reload the information from the server. Updates to this list are done in background without interrupting the work of the user.

4.1.4. "Scale Out" or "Horizontal Scaling"

"Horizontal Scaling" is implemented at two different architecture levels:

- The Open-Xchange application level
- The back-end services level

4.1.4.1 Open-Xchange Application

The application itself is cluster capable. This allows the distribution of the computing load of the Open-Xchange application to many servers running in parallel. There is no limit to how many servers are used in parallel and the cluster can be enhanced and extended transparently and easily.

4.1.4.2 Back End Services

The vertical scalability of the back-end services is a little trickier. The database back-end services can easy become a performance bottleneck because of the physical limits of the database machines. Also the size of the data within each database and table can result in large amounts of data being stored slowing down performance. Methods to minimize these problems are outlined below.

4.1.4.2.1. Read-Write Separation

One trivial but nevertheless very efficient way to enhance the database back-ends scalability is to separate write access to the database from the read access. The Open-Xchange application uses a disproportionate number of read operations when compared to write access. To account for the large number of read operations one or more dedicated read-only databases are created which mirror the data in the "master" write database. Access to the read databases (aka slaves) is balanced with load balancing mechanisms helping to improve both read and write times

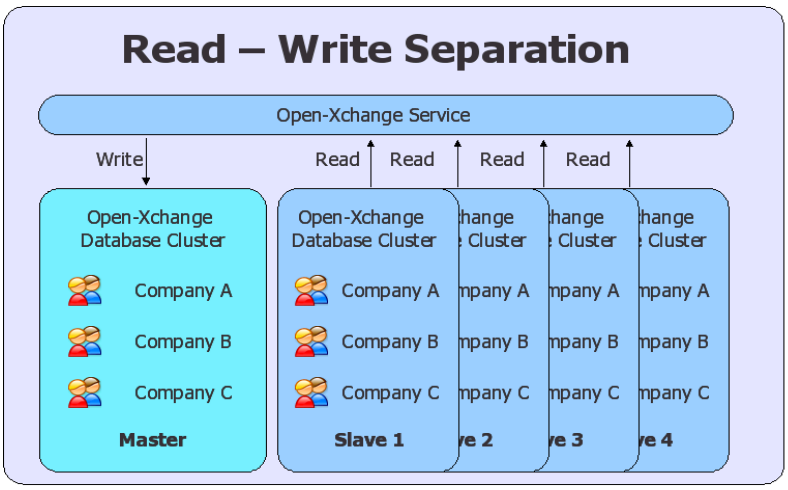

Figure 4: Open-Xchange - Read-Write Separation

Separation of read/write operations allows the installation of the database system in a cluster, with one master and many slaves. The databases use internal replication mechanisms to ensure that all slaves contain the same content as the master database.

All read accesses is sent to the slaves allowing a single database to scale to many more customers and users than a single database handling all read and write operations.

The combination of one master and several slaves containing the same data is called **database cluster** throughout the following chapters.

4.1.4.2.2. Physical Database Partitioning

Even with the read/write separation described above the size of the content in one database will become a limiting factor for the installation when the numbers of customers and users grow extremely large.

To ensure, that the size of a single database server does not grow too large and stays below a pre-defined size, Open-Xchange Server 6 is able to work with a partitioned database setup.

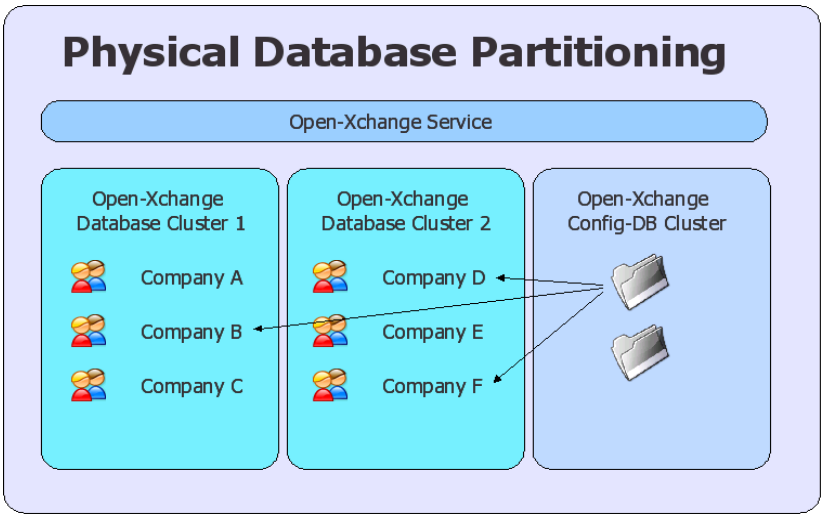

Figure 5: Open-Xchange - Physical Database Partitioning

Physical partitioning means nothing more than distributing the customer data to different database servers/clusters. For example, all data for the first ten thousand customers is stored on database cluster 1 and all data from the second ten thousand customers is stored on database cluster 2.

Additionally, there is a single central database, called "Config-DB", which keeps the meta information of which database cluster contains which customer data. This information is only accessed once during customer login so the Config-DB should not effect scalability of the whole system.

As a result of this architecture, administrators are able to regulate the load on each of the database clusters. Monitoring the load during production provides information on the best way to distribute the customers data into database clusters.

4.1.4.2.3. Logical Database Partitioning

The physical database partitioning described above can be enhanced further into logical database partitioning with Open-Xchange Server 6.

This is useful when the database cluster is able to handle the number of requests but the performance is limited by the size of the database tables. The more a database table grows, the slower the results for each request.

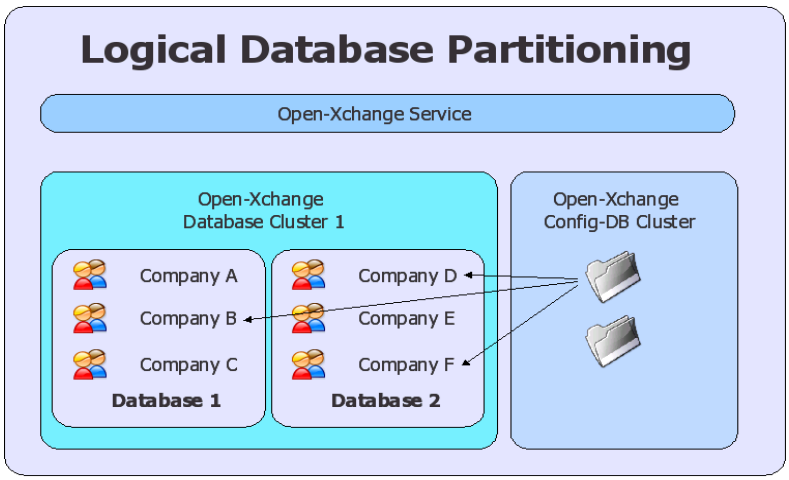

Figure 6: Open-Xchange - Logical Database Partitioning

With this setup it is possible to run several completely separate databases within one database cluster. This allows the same separation of customers into several databases like the physical partitioning except multiple databases run on the same database cluster.

This capability adds yet another level of flexibility to the architecture of Open-Xchange Server 6.

4.2. Multi Tenant / Multi Domain

As described above the Open-Xchange Server 6 can serve many completely separate customer environments within one instance of the Open-Xchange application to efficiently use the existing hardware resources.

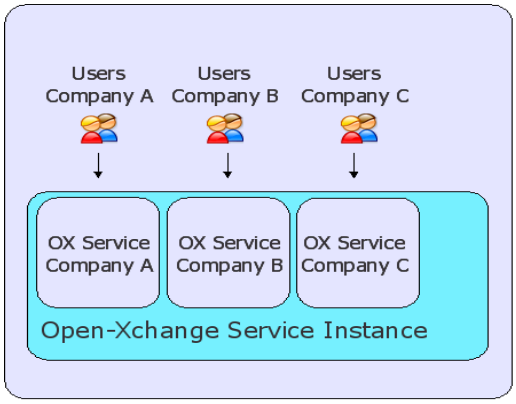

Figure 7: Open-Xchange - Multi Tenant / Multi Domain

Within the context of one customer, only the data for that single customer is available. It is not possible for one user to accidentally or intentionally circumvent this protection and to access data from another customer.

All users within one context work as if they are on a single server dedicated to their installation.

The local data for one context includes everything necessary for the companies to collaborate exactly as if they are running on a dedicated server.

This includes:

- User Administration and Authentication
- Domain Administration and Authentication
- Group Administration
- Resource Administration
- All User Data such as Appointments, Tasks, Contacts, ...

4.3. Integration

Open-Xchange Server 6 is designed to integrate into the existing infrastructure of a hosting provider. Existing services are reused wherever possible.

This is not just limited to integration in the network infrastructure but also applies to business related services like the HR-Software and more.

These integration capabilities adhere to a service oriented approach and are focused on the integration of several different services.

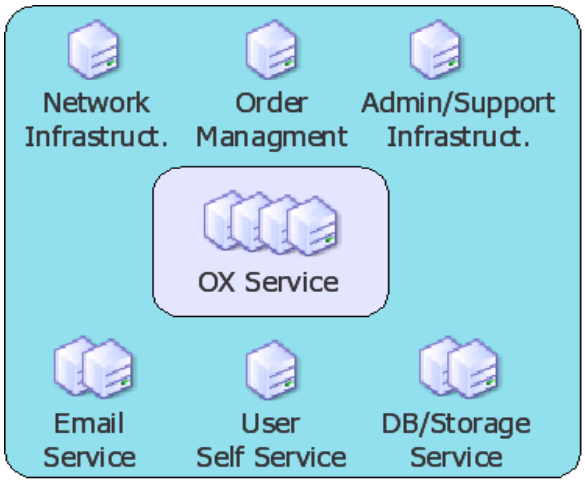

Figure 8: Open-Xchange - Integration

The integration points of note include:

- **HR-Software**
  Existing processes for user deployment and HR-Software will be transparently integrated with the administration framework from Open-Xchange. Integration is based on the standard Java protocol/interface called Remote Method Invocation (RMI) and allows the complete administration of contexts, users, groups and resources.
  When a customer buys the Open-Xchange service, or additional users, the fulfillment of the request is automated and forwarded to the Open-Xchange service. All necessary internal actions such as distribution of the user information between the database clusters will be handled transparently by Open-Xchange Server 6.
- **Email Service**
  Open-Xchange makes use of the Internet standards IMAP and SMTP. All Email systems based on these standards can easily be integrated into an Open-Xchange environment without the need to change anything. This

allows Open-Xchange to be added as a value added service on top of already existing Email offerings.

- **Network Infrastructure**
  Open-Change Server 6 is designed to integrate smoothly in existing network infrastructures with established systems for load balancing and more.
- **Administration Infrastructure**
  Open-Xchange Server 6n is designed to integrate into existing administration infrastructures.
  Administrators can automate the deployment and configuration of Open-Xchange servers. All necessary configuration work can be done remotely, without physical access to the machines. Command-line tools can easily be integrated into automated administration processes.
- **User Self Service**
  Usually a hoster has implemented own web frontends for the user self service. Typically a user can administer his password, vacation notices, mailfilters and more. Open-Xchange offers the so called "Config-Jump", which allows the user to open this web frontends without additional authentication. To achieve that functionality, a plugin will be implemented, which creates a session on the web frontend server for this user.
- **Authentication**
  The authentication process can be implemented as plugin to integrate flexible into the hosters existing authentication infrastructure. It is possible to use flexible login names and to integrate already existing central authentication systems.

4.4. Data Center Integration

Another important aspect of scalability is the ability to flexibly integrate into modern data center processes.

This includes several important requirements:

- high availability – necessary to meet the SLAs of hosting providers
- load balancing – must be flexible to allow for different clustering scenarios
- health monitoring - health issues must be reported fast and monitoring must be reliable to avoid critical situations. Monitoring must also include relevant performance data for tuning the system.

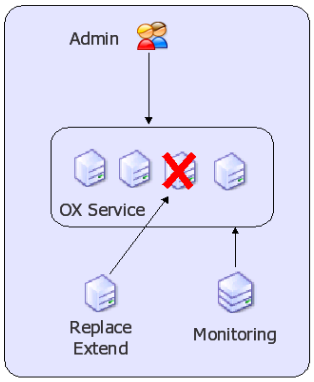

Figure 9: Open-Xchange - Data Center

4.4.1. Cluster Manageability

As described above, Open-Xchange Server 6 is designed to run in a load balanced cluster. To enhance the manageability in this kind of clustered environment there are many notable features which allow for easy administration.

The following concepts are implemented in Open-Xchange Server 6 to provide data center ready operation:

- **Failover for High Availability**
  If one machine breaks down it can be removed from the load balancing mechanism without the other servers in the cluster being informed or reconfigured
- **Enhancement of the cluster**
  Clusters have the ability to be enhanced without the interruption of service. It is possible to add one or more servers transparent to the rest of the cluster and without changing the local configuration on any of the servers. The new server only needs to be registered in the central Config-DB. The same applies for removing one or more machines from the cluster, for example after introducing more powerful servers into the cluster.
- **Enhancement of Back-End Services**
  All Back-End services are transparently enhanced or added to the system. For example, a new database cluster for user data or a new file store for files can be installed and needs only to be registered in the central Config-DB. The next new context will automatically be created in this new database cluster or file spool if appropriate

This flexibility is accomplished through:

- **Identical Configuration**
  All Open-Xchange application servers are configured the same way. This allows easy cloning of the installation and is the basis for the actions described above.
- **Stateless Servers**
  The servers don't rely on any state information from any other servers in the cluster. When users log in they will be assigned to a dedicated Open-Xchange application server. From that point in time, all their requests will be sent to the same server completely independent of the other servers. `mod_proxy_ajp` allows for this type of reliable session behavior.
- **Multicast Auto-configuration**
  There are a few subsystems which need to be aware of the operations of other servers. These include the cache invalidation mechanism for JCS and the Outlook© push trigger mechanism. Both services communicate with their partners in the cluster automatically via multi-cast operations. Using this mechanism it is not necessary to keep hard-coded configuration data in any configuration files and keeps the architecture dynamic.

4.4.2. Monitoring

To run a sophisticated system like Open-Xchange, with many servers and many back end services in a data center, it is mandatory to monitor the health of the entire application stack.

Open-Xchange Server 6 comes with very detailed monitoring capabilities. Everything necessary to interpret the state of the application and its dependencies is monitored.

All the statistics are stored internally and made available via a Java JMX API. This allows Open-Xchange Server 6 to integrate into most monitoring frameworks.

Additionally all of the monitored information can be exported with command line tools and integrate into any other monitoring tool which can parse the text.

5. Frontend Customization

5.1. Themeability

Open-Xchange Server 6 will support theming functionality, so that you will be able to create own themes and make them available in the configuration frontend of the Open-Xchange Webinterface. The stylesheet files are modified, so that everything that should affect the theming functionality can now be configured within an own theme. Besides also the images can be replaced per theme.

All themes will be stored in the following directory of the Open-Xchange Server: `/var/www/ox6/themes/`

5.2. Plug-in Concept for the User Interface

With Open-Xchange Server 6 it is possible to add you own JavaScript Code for existing events. The following areas at the user interface will be plug-in-enabled:

- Administration of groups and resources in the user options
- Special "Config jump" to external control centers
- Displaying further option sites for different services or applications
- Using all contact information in the contact module for further 3rd Party Software
- Activate the SIEVE-Mailfilter functionality at the product

5.3. UWA-Support

Netvibes UWA is a abbreviation for the *Universal Widget API* and describes a standard to make widgets available on every platform such as Netvibes, Apple Dashboard, Blog Systems and many others. With the Open-Xchange Server 6 it is possible to embed those widgets to the User Interface. Additional it is possible to add locally installed UWA Widgets without the connection to NetVibes.

6. Open-Xchange Interfaces

This section describes some of the interfaces offered by Open-Xchange Server 6, which allow access to the Open-Xchange services and data for external applications.

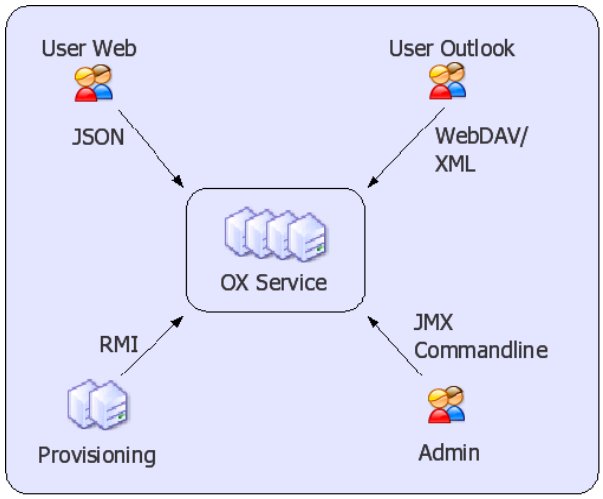

Figure 10: Open-Xchange - Interfaces

6.1. Groupware Data Functions – Java Script Object Notation

The AJAX web front end communicates with the Open-Xchange Server 6 using Java Script Object Notation (JSON).This is the preferred interface to access user's data from external applications

This interface covers all functionality provided by the web front-end including calendar, contacts, tasks, folder/permission management, conflict handling, searches and many more.

"JSON" is a lightweight data-interchange format It is easy for humans to read and write. It is easy for machines to parse and generate. It is based on a subset of the JavaScript Programming Language. JSON is a text format that is completely language independent but uses conventions that are familiar to programmers of the C-family of languages, including C, C++, C#, Java, JavaScript, Perl, Python, and many others. These properties make JSON an ideal data-interchange language."
(http://www.json.org/)

6.2. Groupware Data Synchronization – OXDAV – WebDAV/XML

The OXDAV interface is used for the communication between the Open-Xchange OXtender for Microsoft Outlook© and the Open-Xchange Server 6.

It is based on XML structures, containing the commands and the resulting data.

This XML objects are transferred through the internet protocol WebDAV, which is based on http.

This interface offers all functionality which is necessary for the Open-Xchange OXtender for Microsoft Outlook© to work properly. Therefore it is the best protocol for synchronizing complete data sets with external applications (i.e. rich clients).

6.3. Provisioning – Remote Method Invocation (RMI)/SOAP

The Open-Xchange administration framework communicates via Java Remote Method Invocation or SOAP(Java RMI). This interface allows execution of all the administrative actions on the Open-Xchange system. This includes context management, user administration, group and resource configuration as well as the registration of the back-ends database cluster and filestore.

The RMI/SOAP interface into the Open-Xchange administration framework is used for the integration into existing user provisioning systems and HR-Software processes.

Any Java based application can use this interface to gain access to administrative functions. The main advantage of this communication interface is the lack of any artificial protocol overhead which results in very high performance.

"Java Remote Method Invocation (Java RMI)/SOAP enables the programmer to create distributed Java technology-based to Java technology-based applications, in which the methods of remote Java objects can be invoked from other Java virtual machines, possibly on different hosts. RMI uses object serialization to marshal and unmarshal parameters and does not truncate types, supporting true object-oriented polymorphism."
(http://java.sun.com/javase/technologies/core/basic/rmi/index.jsp)

6.3.1. Synchronization from LDAP/ADS

The user database of Open-Xchange Server 6 can be synchronized with an existing LDAP Server or Active Directory Server via a special script. The following functionality are possible:

- Create new user

- Modify existing user
- Delete user
- Create group
- Delete group
- Allocate user to group
- Support of all Open-Xchange fields

6.4. System Administration Interfaces

Open-Xchange offers two very important interfaces for System Administrators:

6.4.1. Monitoring – Java Management Extensions

Open-Xchange makes use of the Java Management Extensions (JMX) to provide all runtime statistics and potential errors to monitor the health status of the Open-Xchange Server 6.

This interface can be accessed with any application capable to access JMX information. Additionally Open-Xchange delivers command line monitoring tools, which can be used to output all collected monitoring information as plain text to the console. This can be used for the flexible integration into any monitoring framework.

"Java Management Extensions (JMX) technology provides the tools for building distributed, Web-based, modular and dynamic solutions for managing and monitoring devices, applications, and service-driven networks. By design, this standard is suitable for adapting legacy systems, implementing new management and monitoring solutions, and plugging into those of the future."
(http://java.sun.com/javase/technologies/core/mntr-mgmt/javamanagement/)

6.4.2. Administration – Command-line Tools

In addition to the Java RMI interface, the Open-Xchange administration framework allows access to all relevant functions through command-line tools.

This allows the administrator to do all relevant administration tasks fast and system independent directly on the console. This makes it very easy to fulfill the necessary tasks in the administrator's daily work.

