

LiquidOffice v5 Architecture/Technologies

OVERVIEW	3
<i>Components</i>	<i>3</i>
1. STORAGE	5
<i>1.1. Static Elements.....</i>	<i>5</i>
<i>1.2. Dynamic Elements</i>	<i>7</i>
2. COMMUNICATION	11
<i>2.1. Client-Tier <-> Web Server Tier Communication.....</i>	<i>12</i>
<i>2.2. Web Server Tier<-> Application Tier Communication</i>	<i>12</i>
<i>2.4. Application Tier <-> Data Tier.....</i>	<i>12</i>
3. CLUSTERING.....	12
<i>3.1. Cluster Tag</i>	<i>12</i>
<i>3.2. Front loaded Load Balancing.....</i>	<i>12</i>
<i>3.3. Clustered Process Server Files.....</i>	<i>12</i>
4. CONFIGURATIONS	12
<i>4.1. Default Ports.....</i>	<i>12</i>

Overview

This document is designed to help network/security administrators understand the architecture and technologies used in LiquidOffice. It is important to understand the following information before deploying LiquidOffice – especially to the Internet where security issues are of the highest concern.

The LiquidOffice Server may reside on an internal private Intranet or on the Internet. How the server is made available for use to internal users and external users in either of these configurations will vary. The various configurations will depend on each organization's network architecture and security requirements.

The sections that follow are intended to provide the information necessary to determine how LiquidOffice can best fit into any given network architecture.

Overview

Components

LiquidOffice consists of these main components: Form Designer, Process Studio, Presentation Server, Process Server, Web Desktop, Mobile Client, My Data Client, and Management Console.

1. LiquidOffice Form Designer is used by form designers to create forms and publish them to the server. When installing the Form Designer, the following components are installed to the workstation:
 - a. Form Designer
 - b. Java Runtime Environment version 1.5 (if not already installed)
2. LiquidOffice Process Studio is used by process designers to create processes and publish them to the server. The Process Studio is downloaded from the LiquidOffice Server to the workstation through Java Web Start. The following component must be installed on the workstation to run the Process Studio:
 - a. Java Runtime Environment version 1.5 (which includes Java Web Start)
3. LiquidOffice Server is comprised of the following components:
 - a. LiquidOffice Presentation Server hosts the Management Console, Publishing Wizard, and My Data Client and builds the *gateway* to the Process Server. It hosts and renders the localized Web Desktop dynamically.
 - b. LiquidOffice Process Server hosts the forms and processes published from the Form Designer and Process Studio. The server is responsible for

serving up forms, interacting with the forms for lookups, validations and form submittals including routing. Moreover, the Process Server handles the initiation, management, and tracking of processes while also serving as the Process Engine for enforcing the business rules specified by the process definition.

- c. Database Server for storing active content within the LiquidOffice system, referred to as the 'Internal Database Server' in this document.
 - d. Database Server for storing all historical transactions for reporting and auditing purposes, referred to as the 'Archive Database Server' in this document (optional).
 - e. LDAP Server (optional).
 - f. Blackberry Enterprise Server (optional).
4. LiquidOffice My Data Client provides end users direct access to exported data, e.g. for importing into Microsoft Office.
 5. LiquidOffice Web Desktop provides access to published forms and processes, and supplies the user interface for routing these forms and processes. A standard web browser is used to display the Web Desktop. The following components are needed to run the Web Desktop:
 - a. Web Browser
 - b. Acrobat or Acrobat Reader (only needed to view PDF forms)
 6. LiquidOffice Mobile provides access to in-route forms and processes, and supplies the user interface for routing these forms and processes. A mobile device is used to run LiquidOffice Mobile. Currently, the Blackberry is the only supported mobile device. The following components are needed to run LiquidOffice Mobile:
 - a. Blackberry Device running OS 4.0 or higher
 7. LiquidOffice Management Console provides administration capabilities for all aspects of the server. The Management Console is downloaded from the LiquidOffice Server to the workstation through Java Web Start. The following component must be installed on the workstation to run the Management Console:
 - a. Java Runtime Environment version 1.5 (which includes Java Web Start)

1. Storage

1.1. Static Elements

The elements installed to support each of the above listed components are as follows:

1. LiquidOffice Form Designer is a component-based architecture wherein each component becomes visible within the UI if it is present on the system. The install program places each component into the directory chosen at install time. The Form Designer can be installed by an Administrator and used by any user locally on that machine. The Form Publishing Wizard in the Form Designer is launched through the use of Java Web Start. The first time the Publishing Wizard is used, Java Web Start will download the wizard from the LiquidOffice Server. The Form Designer is only supported on Windows.
2. LiquidOffice Process Studio is available for download from the LiquidOffice Server through the use of Java Web Start (Studio Icon in the Web Desktop or via URL, <http://<LiquidOfficeServer>/studio>). The JRE 1.5 is needed to run the Process Studio; if launching the Process Studio from the Web Desktop, a script will be run to detect if the client system has the JRE 1.5 installed. If the JRE is not detected the user will be prompted to install the JRE from the LiquidOffice Server. The Process Studio is supported on Windows and Macintosh.
3. LiquidOffice My Data Client is available for download from the LiquidOffice Server (Install button in the Web Desktop's My Data Tab). My Data Client implements CSV, Delimited, Excel and Access exports without the use of ODBC and so does not require that MDAC be installed. It does require the presence of Excel or Access on the system that uses one of those data export options. The My Data Client is only supported on Windows.
4. The Web Browser is expected to be installed by the user. LiquidOffice does not ship with any web browsers. The user is expected to have Acrobat or Acrobat Reader installed on their system to use forms published as PDF. PDF forms are displayed inside the browser using the PDF ActiveX control, or Plug-in. The Web Desktop is supported on Windows and Macintosh OS X. See the Operating System Wizard section of the LiquidOffice support web site at https://customers.verity.com/support/secure/os_wiz.jsp for the versions of web browsers and Acrobat that are supported for each operating system:
 - a. Internet Explorer, Firefox, Safari
 - b. Acrobat and Acrobat Reader
5. LiquidOffice Mobile is an add-on option available for purchase. If purchased, LiquidOffice Mobile features will be enabled. Once enabled, users will be able to send the LiquidOffice Mobile application to their Blackberry device from the Web Desktop. LiquidOffice Mobile may also be administratively installed by a Blackberry Enterprise Server administrator. The user is

expected to have a previously configured Blackberry device running OS 4.0 or higher. LiquidOffice Mobile is only supported on Blackberry devices.

6. LiquidOffice Management Console is available for download from the LiquidOffice Server through the use of Java Web Start (Administration Icon in the Web Desktop or via URL, <http://<LiquidOfficeServer>/lomc>). The JRE 1.5 is needed to run the Management Console; if launching the Management Console from the Web Desktop a script will be run to detect if the client system has the JRE 1.5 installed. If the JRE is not detected the user will be prompted to install the JRE from the LiquidOffice Server. The Management Console is supported on Windows and Macintosh.
7. The LiquidOffice Server ships with everything it needs except the operating system, the database engines (see Database Server below), an LDAP server (see LDAP Server below), and a Blackberry Enterprise Server (see BES Server below). The Administrator must perform the install on a Solaris, Linux or Windows Server machine. Components shipped with the LiquidOffice Server include:
 - a. LiquidOffice Process Server
 - i. J2SE v1.5 JRE and other classes from Sun Microsystems, Inc.
 - ii. Tomcat Servlet Container from the Apache Software Foundation
 - iii. LiquidOffice Application Files (Java classes, JSP pages and configuration files)
 - iv. K2 VDK
 - v. DataDirect MS-SQL driver
 - vi. Rhino script engine from the Mozilla Foundation
 - vii. AXIS from the Apache Software Foundation
 - viii. BeanShell (<http://www.beanshell.org/>)
 - ix. InkToolsLib from Communication Intelligence Corp.
 - x. Libraries from ID Automation for barcode support (<http://www.idautomation.com/>)
 - b. LiquidOffice Presentation Server
 - i. J2SE v1.5 JRE and other classes from Sun Microsystems, Inc.
 - ii. Tomcat Servlet Container from the Apache Software Foundation

- iii. LiquidOffice Application Files (Java classes, JSP pages and configuration files)

A complete list of all libraries included in LiquidOffice can be found here:
<http://<LiquidOfficeServer>/doc/libraries.html>

- 8. The Database Server(s) are not included with LiquidOffice. The following are supported:

- a. Microsoft SQL Server 2000 SP4
- b. Microsoft SQL Server 2005
- c. Oracle 9i
- d. Oracle 10g

If Microsoft SQL Server 2005 or Oracle 10 is not selected for the Archive database, some reporting functionality may be unavailable.

- 9. The LDAP Server is not included with LiquidOffice. Administrators can create/import their own users or choose to connect to one or more of the following LDAP servers:

- a. Microsoft Active Directory 2003
- b. SunOne/iPlanet Directory Server 5.2
- c. Novell eDirectory 8.7
- d. Oracle Internet Directory 10

- 10. The Blackberry Enterprise Server (BES Server) is not included with LiquidOffice. Administrators are expected to have a configured BES Server running Mobile Data Services in order to use LiquidOffice Mobile.

1.2. Dynamic Elements

The elements handled by each of the above listed components at run time are as follows:

- 1. Form Designer stores natively the forms it creates as an XML document. A published version of a form contains additional information. The information that may be contained in a published form is listed here:
 - a. Form in Native XML format
 - b. Form data representation as XML definition for server. Contains field definitions, database validation and lookup definitions, form script for server entry points.

- c. PDF representation of form including form script for client entry points
- d. HTML representation of form including form script for client entry points
- e. Current script implementation of LiquidOffice Java Object Model (LJOM) for use with HTML

Forms may be stored in local or network directories and when design environments are shared, designers may use source control tools externally to the application to coordinate access to form definitions.

Form Designer stores information about cached login credentials and user preferences in the Windows Registry.

2. Process Studio stores natively the process definitions it creates as an XML document. Process definitions may include custom script code and be stored in local or network directories and when design environments are shared, designers may use source control tools external to the application to coordinate access to process definitions.

Process Studio stores information about cached login credentials and user preferences locally in

```
<user home directory>/LiquidOffice/studio.xml
```

3. My Data Client stores information about exports done through the My Data Client. Information stored includes the format last 'exported to' for each form as well as the column mappings, if any. This information is stored in the Documents and Settings area for the user.
4. Web Browser stores a cookie for the user with information about web desktop preferences. This information is refreshed from the server once the user logs in. The browser is required to accept a cookie in order to allow the user to log in to the LiquidOffice server.

The Web Browser displays and may cache the following information:

- a. Web Desktop pages (Inbox, Forms Repository, etc.)
 - b. Blank Forms
 - c. Attachments
5. LiquidOffice Mobile stores mobile user preferences as well as form data locally on the mobile device in native format. All of the stored data is deleted when the application is removed from the device.

6. Management Console stores information about cached login credentials and user preferences locally in
`<user home directory>/.LiquidOffice/config/.`

7. The physical disk where the LiquidOffice Presentation Server is installed contains the following dynamic data:

Settings configured in the web application's deployment descriptor (WEB-INF/web.xml), such as the path to LiquidOffice Process Server. This is not really dynamic since it is typically configured once and not touched after that.

8. The physical disk (Shared Resources Folder) where the LiquidOffice Process Server is installed contains the following dynamic data:

- a. Published forms (including client and server script)
- b. Published processes (including server script)
- c. Published reports
- d. Some submitted forms (where a digital signature is involved)
- e. Form attachments
- f. Process attachments
- g. Generated reports
- h. VDK Indexes for search functionality
- i. Connect Agent export scripts
- j. Settings captured during install such as database configuration, SSL configuration, etc. The database password is stored on disk in an encrypted format.

9. The Internal Database Server contains all other dynamic information including:

- a. References to all published forms, processes and reports
- b. All data export mappings and other defined form and process properties
- c. References to all form and process attachments
- d. All submitted data, notes and route traces for active events
- e. All process templates and process instance data

- f. Every event in a user's Inbox, Saved, Sent and Deleted items folders
 - g. All user information including profile fields. If LDAP is not being used, the user's password is encrypted and stored in the database using an MD5 hash since decryption is never necessary. If using LDAP, the user's password is never stored.
 - h. All user form, process and report permissions
 - i. All queued e-mails
 - j. All Connect Agent setup information including lightly encrypted passwords needed to connect to the database server. (Passwords are lightly encrypted so they can be decrypted and used to access a connect agent)
 - k. All Administration-related information such as server settings, folder definitions, etc.
10. The Archive Database server contains all transaction history information including:
- a. References to all submitted forms and processes (in route or completed)
 - b. All submitted data (in route or completed)
 - c. References to all administrative actions (user added, form published, etc.)
11. The LDAP Server contains all dynamic user attributes and is synchronized with the internal database server at intervals specified by the LiquidOffice administrator.

2. Communication

The communication between all involved components can be best understood by looking at the underlying LiquidOffice 4-tier architecture.

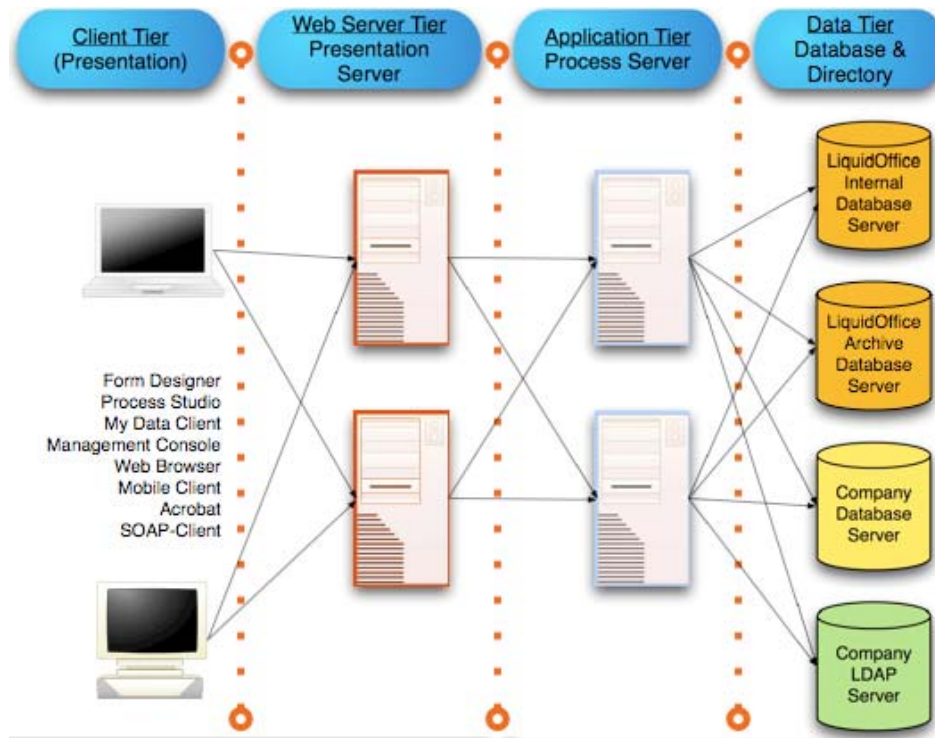


Figure 1: LiquidOffice 4-tier Architecture

All client communication takes place between a client (Form Designer, Process Studio, My Data Client, Management Console, Web Browser, Mobile Client, Acrobat, SOAP-Client) and the LiquidOffice Presentation Server.

The Presentation Server may respond to requests directly but typically consults with the LiquidOffice Process Server before rendering a response.

All database communication takes place between the LiquidOffice Process Server and the Database Server(s).

All directory lookups and searches take place between the LiquidOffice Process Server and the LDAP Server.

Clients never communicate directly with the LiquidOffice Process Server, Database Server(s) or the LDAP Server. Generally speaking, only components in adjacent tiers communicate directly.

The communication between all components (Form Designer, Process Studio, My Data Client, Management Console, Browser, Mobile Client, LiquidOffice Presentation Server, LiquidOffice Process Server, Database Server(s), and LDAP Server) is discussed below.

2.1. Client-Tier <-> Web Server Tier Communication

The following sections describe how components in the Client Tier, namely Form Designer, Process Studio, My Data Client, Management Console, Mobile Client, and Web Browser communicate with components in the Web Server Tier, namely the LiquidOffice Presentation Server. Depending on the network topology, there may be additional devices like Firewalls and Load-Balancers between the two tiers (see 2.1.6-Optional-Components for details).

Intra-tier communication does not happen in the Client Tier and happens only in the Web-Server-Tier, if the Presentation Server is clustered (see 2.2.5-Intra-Tier-Communication for details).

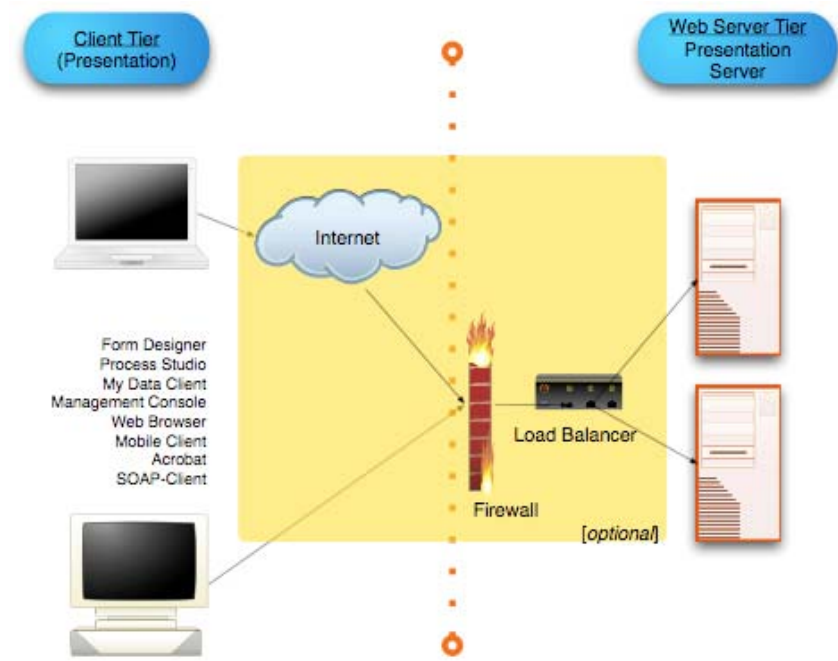


Figure 2: Client-Tier <-> Web-Server-Tier

2.1.1. Form Designer <-> LiquidOffice Presentation Server

Form Designer communicates with the LiquidOffice Presentation Server (see Figure 2) during form publishing and in some cases during form design. Events that trigger server communication during design include:

- a. Opening the script editor
- b. Setting up database validations
- c. Setting up database lookups

- d. Setting up user profile field mappings
- e. Setting up list boxes that get pre-filled from a database

Whenever communication with the LiquidOffice Server is necessary, the Form Designer prompts the user for login information. The login information is used to maintain communication with the server until the user logs out (possibly by shutting down Form Designer) or the server session expires. The expiration time is configured in a Web Application's deployment descriptor:

```
WEB-INF/web.xml : web-app/session-config/session-timeout
```

Form Designer offers the user the opportunity to remember the password for convenient logging in next time. If the user selects this option, the password gets stored in the registry in an encrypted form.

When publishing a form, all of the elements listed in 1.2-Dynamic-Elements, are sent to the LiquidOffice Server.

When setting up database validations, lookups, etc., LiquidOffice Server sends the following information to the Form Designer:

- a. List of Connect Agent names
- b. List of table names served by a specified Connect Agent
- c. List of column names that are part of a specified table
- d. List of User Profile fields defined on the server

All communication between Form Designer and LiquidOffice Server is done by sending XML commands and XML data via HTTP or HTTPS (over TCP/IP). No other protocols (i.e. RPC or NetBios) are used or required. Data (including user name and password) is sent in clear text via HTTP or encrypted if via HTTPS. Typically the LiquidOffice Presentation Server listens on port **80** for HTTP and **443** for HTTPS. Both ports are configurable during installation of the LiquidOffice Server.

If the LiquidOffice Presentation Server is deployed in Tomcat, it keeps the mentioned port configuration in the server configuration file:

```
$CATALINA_BASE/conf/server.xml : Server/Service/Connector.port
```

2.1.2. Process Studio <-> LiquidOffice Presentation Server

Process Studio communicates with the LiquidOffice Server (see Figure 2) during process publishing and in some cases during process design. Events that trigger server communication during design include:

- a. Browsing for forms, processes or users

- b. Setting process field properties and values

Whenever communication with the LiquidOffice Server is necessary, the Process Studio prompts the user for login information. The login information is used to maintain communication with the server until the user logs out (possibly by shutting down Process Studio) or the server session expires. The expiration time is controlled in a Tomcat setting.

Process Studio offers the user the opportunity to remember the password for convenient logging in next time. If the user selects this option, the password gets stored in encrypted form here: `<user home directory>/ .LiquidOffice/studio.xml`

When publishing a process the Process Definition is sent to the LiquidOffice Server. When setting up Form-Edit tasks, etc. LiquidOffice Server sends the following information to the Process Server:

- a. List of folders
- b. List of forms
- c. List of fields for a form
- d. List of users

All communication between Process Studio and LiquidOffice Presentation Server is done by sending XML commands and XML data via HTTP or HTTPS (over TCP/IP). Data (including user name and password) is sent in clear text via HTTP or encrypted if via HTTPS. The same ports are used for Form Designer and Process Studio communication.

2.1.3. My Data Client <-> LiquidOffice Presentation Server

When a user chooses 'Get All Data' or 'Get New Data' from the Web Desktop, the My Data Client is launched on the local PC. The My Data Client communicates with the LiquidOffice Server using the same session the browser is using and therefore does not require further login (see Figure 2).

LiquidOffice Server sends information about the form selected (field names, types, etc.) to the My Data Client. My Data Client stores this information locally for future use. LiquidOffice Server also sends the requested data records to the My Data Client. These records are then exported to the specified local database or file.

All communication between My Data Client and LiquidOffice Presentation Server is done by sending XML commands and XML data via HTTP or HTTPS (over TCP/IP). Data is sent in clear text via HTTP or encrypted if via HTTPS. The same ports are used for Form Designer, Process Studio and My Data Client communication.

2.1.4. *Web Desktop <-> LiquidOffice Presentation Server*

When using the Web Desktop, the Browser communicates with the LiquidOffice Presentation Server for each page in the UI that is displayed (see Figure 1). All communication between Web Desktop and LiquidOffice Presentation Server is done by sending XML commands and XML data via HTTP or HTTPS (over TCP/IP). Data is sent in clear text via HTTP or encrypted if via HTTPS. The same ports are used for Form Designer, Process Studio, My Data Client and Web Desktop communication.

When PDF forms are viewed and submitted, both viewing and submission is done within the browser and all communication is via standard HTTP or HTTPS. In order to allow the user to log in (and be kept logged-in) to the server and have consecutive requests sent to the same server, the web browser is required to accept cookies. Non-routing forms that have also been configured to allow anonymous access however do not require a user to log in and do not require the browser to accept cookies.

2.1.5. *Blackberry Client <-> LiquidOffice Presentation Server*

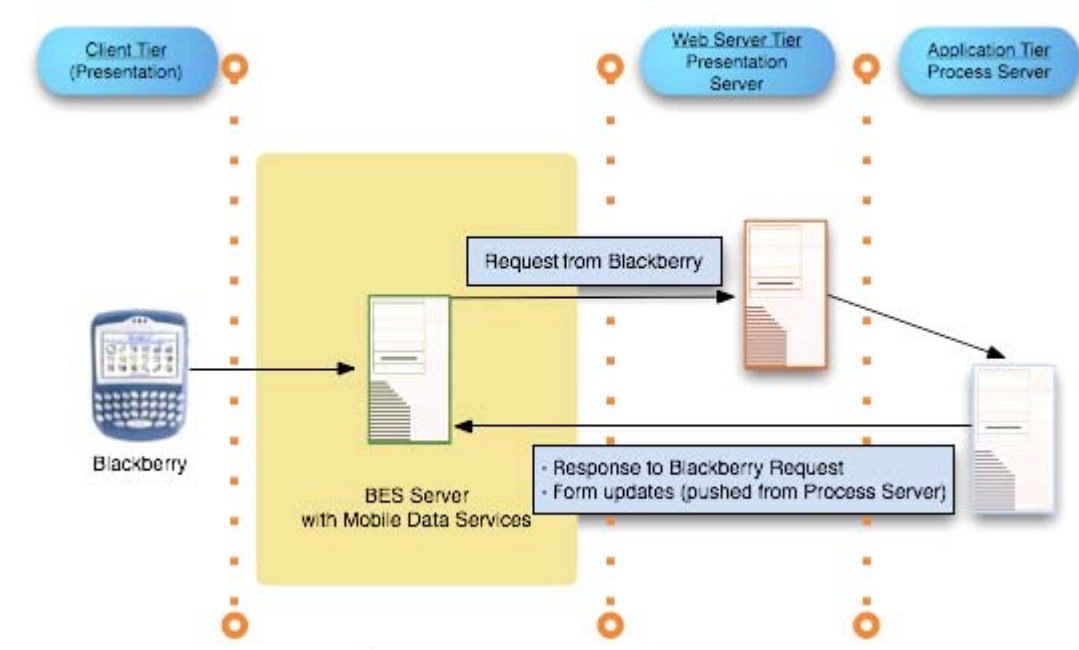


Figure 3: Blackberry Communication

In order to receive in-route forms on a Blackberry Client, the user must enable the server to route forms to their device. This is done via the Web Desktop User Profile Tab. Each device is mapped to a single user via an email-to-device-pin mapping stored on the Blackberry Enterprise Server (BES). Only users in the LiquidOffice Mobile role are given this option in the Web Desktop.

When using the Blackberry Client, all requests from the device communicate with the LiquidOffice Presentation Server, using the Mobile Data Services (MDS) plugin on the BES Server as a proxy (see Figure 3). Each form that is submitted from the device initiates communication with the LiquidOffice Presentation Server. All responses to Blackberry client-initiated requests are sent directly from the LiquidOffice Process Server to the BES Server. The LiquidOffice Process Server also pushes form updates directly to the BES Server.

Communication between the Blackberry Client and the LiquidOffice Presentation and Process Servers is done by sending XML commands and XML data via HTTP or HTTPS (over TCP/IP).

Communication between the Blackberry device and the BES Server is encrypted using tripleDES encryption. Communication between the BES Server and the LiquidOffice Presentation and Process Servers is sent in clear text via HTTP or encrypted if via HTTPS. The same ports are used for Form Designer, Process Studio, My Data Client, Web Desktop, and Blackberry Client communication.

In order to allow the user to log in to the server, a single-use token is sent with each form. Once used, the token becomes invalid and may not be used a second time. At any time a user may disable their Blackberry via the Web Desktop User Profile Tab. After this, no forms will be routed to or accepted from the user's device.

2.1.6. Management Console <-> LiquidOffice Process Server

The Management Console communicates with the LiquidOffice Presentation Server (see Figure 2). Like all other components in the client tier, LiquidOffice Management Console communicates with the LiquidOffice Presentation Server by sending commands and parameter data via HTTP or HTTPS (over TCP/IP). The server responds with XML documents sent over HTTP or HTTPS. The aforementioned ports are used.

Management Console offers the user the opportunity to remember the password for convenient logging in next time. If the user selects this option, the password gets stored in encrypted form here: `<user home directory>/ .LiquidOffice/config/`

When PDF forms are viewed and submitted, both viewing and submission is done within a browser and all communication is done via standard HTTP or HTTPS.

2.1.7. SOAP

SOAP clients are no exception in communicating with the LiquidOffice Presentation Server (see Figure 2). Like with all other components in the client tier, communication is via HTTP or HTTPS (over TCP/IP). The following URL can be used to test if the LiquidOffice server has the SOAP interface deployed:

<http://<PresentationServer>/xmlserver/services>

A list containing three SOAP services including their methods is returned in case the SOAP interface has been deployed:

- Version
- LOService
- LOServiceV3

2.1.8. Optional Components between Client and Web-Server Tier

Firewall and Load-Balancer are two of the components most likely to be found between the Client-Tier and the Web-Server-Tier. Hosting public-facing forms is just one of the reasons why the LiquidOffice Presentation Server needs to be accessible from the Internet, which may trigger the need for a Firewall.

The Blackberry Enterprise Server (BES) also falls into this category. The BES acts as a proxy between the LiquidOffice Presentation Server and the Blackberry Client devices allowing the devices to communicate with the LiquidOffice Presentation Server as if they were on the local network.

In case the LiquidOffice Presentation Server is run clustered, a Load Balancer can be found between the Client-Tier and the Web-Server-Tier (see Figure 2 and section 2.2.4.).

2.2. Web Server Tier<-> Application Tier Communication

The following sections describe how components in the Presentation Tier, namely the LiquidOffice Presentation Server, communicate with components in the Application Tier, namely the LiquidOffice Process Server and Process Engine.

Depending on the network topology, there may be additional devices like Firewalls and Load-Balancers between the two tiers.

If components in a tier are clustered, there will also be intra-tier communication (see 2.2.5-Intra-Tier-Communication for details).

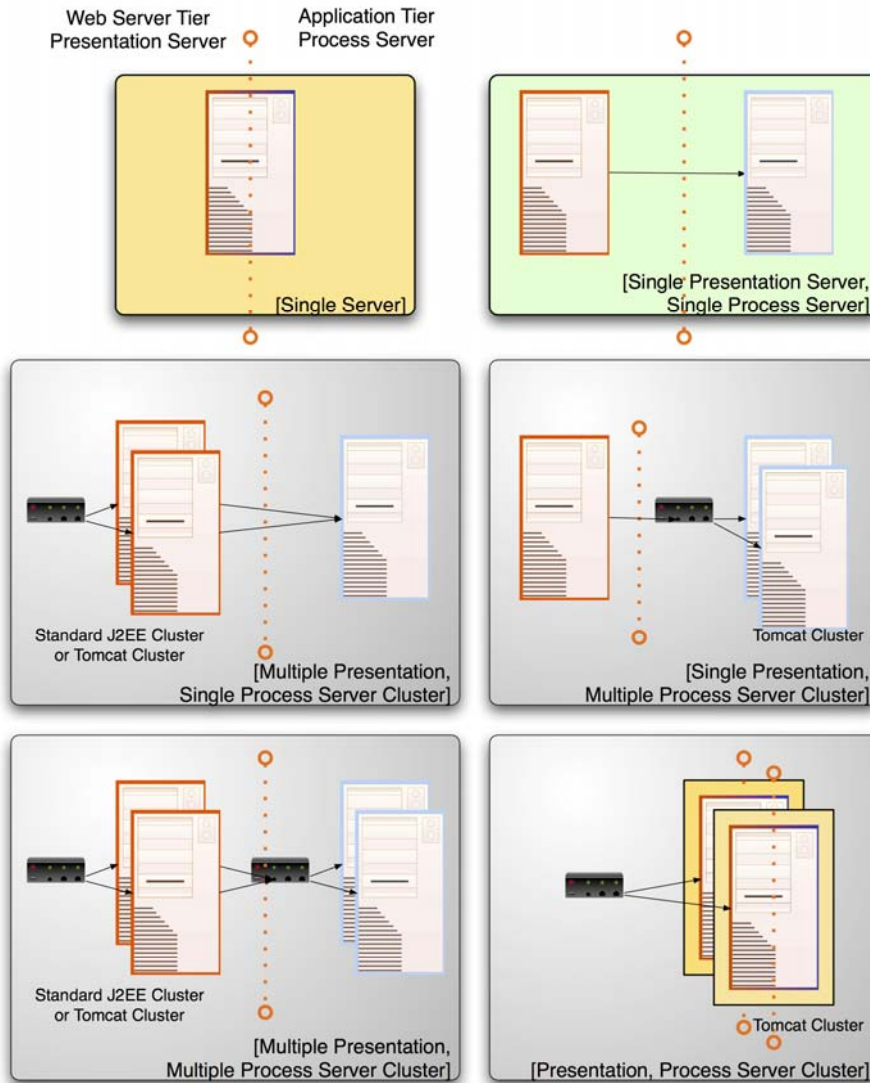


Figure 3: Web-Server-Tier <-> Application-Server-Tier

2.2.1. Two Contexts on a Single Server

A LiquidOffice Server install always deploys both server components into a single Tomcat Servlet container. By default, LiquidOffice Presentation Server is deployed as the **ROOT** context on port **80**, while the Process Server is deployed into the **/xmlserver** context on the same port.

The related protocol and port information is kept in the server configuration file: `§CATALINA_BASE/conf/server.xml`.

After a default installation the CATALINA_BASE environment variable would have been defined like this:

Windows: CATALINA_BASE=C:\Program Files\Cardiff\LiquidOffice\xmlserver\xmlbase

Solaris: CATALINA_BASE=/usr/local/cardiff/liquidoffice/xmlserver/xmlbase

Linux: CATALINA_BASE=/usr/local/cardiff/liquidoffice/xmlserver/xmlbase

In this case, the server configuration file is found here:

Windows: C:\Program Files\Cardiff\LiquidOffice\xmlserver\xmlbase\conf\server.xml

Solaris: /usr/local/cardiff/liquidoffice/xmlserver/xmlbase/conf/server.xml

Linux: /usr/local/cardiff/liquidoffice/xmlserver/xmlbase/conf/server.xml

The context descriptors for all deployed contexts are located here:

Windows:
C:\Program Files\Cardiff\LiquidOffice\xmlserver\xmlbase\conf\Catalina\localhost\

Solaris:
usr/local/cardiff/liquidoffice/xmlserver/xmlbase/conf/Catalina/localhost

Linux:
usr/local/cardiff/liquidoffice/xmlserver/xmlbase/conf/Catalina/localhost

- root.xml – describes the Presentation Server context
- xmlserver.xml – describes the Process Server context
- static.xml – describes the context for static resources, e.g. the built-in demo forms are stored in this context.
- lopinstall.xml – describes the context hosting the LiquidOffice Presentation Server installer for Windows, Solaris and Linux.
- loextras.xml – describes the context hosting the installable resources themselves (loserver.war and tomcat specifically).

A context descriptor simply points to the location on a local disk or **shared network drive**, where the content or Web Application is located. The complete context descriptor for the LiquidOffice Process Server for instance, may look something like this on

Windows:

```
<?xml version="1.0" encoding="UTF-8"?>
<Context path="/xmlserver" docBase="C:\Documents and Settings\All
Users\Application Data\Cardiff\LiquidOfficeServer\SharedResources\xmlserver"
debug="0" reloadable="false" crossContext="true" privileged="true">

    <Manager className="org.apache.catalina.session.PersistentManager"
saveOnRestart="false">

        <Store className="org.apache.catalina.session.FileStore"/>

    </Manager>

</Context>
```

Solaris:

```
<?xml version="1.0" encoding="UTF-8"?>
<Context path="/xmlserver" docBase="/Liquidoffice/sharedresources" debug="0"
reloadable="false" crossContext="true" privileged="true">

    <Manager className="org.apache.catalina.session.PersistentManager"
saveOnRestart="false">

        <Store className="org.apache.catalina.session.FileStore"/>

    </Manager>

</Context>
```

Linux:

```
<?xml version="1.0" encoding="UTF-8"?>
<Context path="/xmlserver" docBase="/Liquidoffice/sharedresources" debug="0"
reloadable="false" crossContext="true" privileged="true">

    <Manager className="org.apache.catalina.session.PersistentManager"
saveOnRestart="false">

        <Store className="org.apache.catalina.session.FileStore"/>

    </Manager>

</Context>
```

Without specifying the `PersistentManager` in the context descriptor, Tomcat would try to restore sessions after rebooting. (Restarting the server writes a `SESSIONS.ser` file in the `%CATALINA_BASE%\work\Catalina\localhost\xmlserver` folder on shutdown and restores sessions on startup, i.e. if they have not expired in the meantime.) In general, this is a good thing and the Presentation Server context for instance uses this feature. However, since the Process Server's internal LO session map is not regenerated during a restart, restoring gets properly disabled for the `/xmlserver` context by inserting the above shown `<Manager ..>` tag into the context's descriptor.

Moreover, the Context tag's `reloadable` attribute is set to **false**. Letting Tomcat monitor classes in `/WEB-INF/classes/` and `/WEB-INF/lib` for changes, to automatically reload the web application if a change is detected, would require significant runtime overhead and is not recommended for use on deployed production applications.

While Presentation Server and Process Server belong to different tiers, they can still be deployed on the same physical server, running in the same Servlet container. This has the advantage that the communication between the two components does not need to go over the (local-)network. Instead, the Presentation Server addresses the Process Server as `localhost`.

As mentioned before, only components in adjacent tiers are allowed to communicate directly. In that sense, the Presentation Server acts as a client towards the Process Server, always initiating communication between the two.

Four context parameters for host, protocol, port, and context, defined in the Presentation Server's deployment descriptor, configure how the Presentation Server communicates with the Process Server. After a default installation the deployment descriptor would be located here:

```
\\<SharedResourcesDirectory>\lopserver\WEB-INF\web.xml
```

The parameters would look something like this:

- LOXmlSchema = http
- LOXmlHost = localhost
- LOXmlPort = 80
- LOXmlContext = /xmlserver

All communication between the two tiers is done via standard HTTP. Data (including user name and password) is sent in clear text via HTTP.

2.2.2. Deploying the Presentation Server on a separate Machine

After installing LiquidOffice server, both server applications are deployed as two contexts. However, using the /lopinstall context (see 2.2.1) allows the convenient deployment of the Presentation Server on another machine on the same network.

The /lopinstall context offers executable installs for Solaris, Linux and Windows. Both installs include a Java VM, the Tomcat Servlet Container, and the LiquidOffice Presentation Server. During the installation, LiquidOffice Process Server's URL and Port information needs to be entered.

For the convenient deployment of the LiquidOffice Presentation Server into an existing J2EE Application Server or Servlet Container, the Presentation Server has been packaged into a J2EE Web archive (WAR), which can be downloaded from the Process Server's *extras* context:

```
http://<ProcessServer>/xmlserver/extras/lopserver.war
```

E.g., IBM WebSphere 6.1 and BEA WebLogic 9.2 are certified J2EE Application Servers that can be used to deploy the LiquidOffice Presentation Server Web archive.

Depending on the network topology, the HTTP Connector's proxyName and proxyPort attributes may have to be adjusted manually on the Process Server's Tomcat installation (see 3.2-Clustering for details).

2.2.3. *Clustering a Single Server*

The LiquidOffice servers can be clustered in many different ways. However, the simplest and for most applications also most effective clustering would keep the communication between the two server components (Presentation and Process Server) off the network.

In this case each cluster node runs all contexts and inter-context communication only happens locally (through localhost). A load balancing front-end needs to ensure that all incoming requests with the same session are routed to the same node (see 3-Clustering for details).

2.2.4. *Combining all models*

While the practicality may be doubtful, by combining the models described above, more clustering scenarios can be imagined:

2.2.4.1. *Clustered Presentation Server - Single Process Server*

As described in Section 2.2.3, a load balancing front-end distributes the load among several Presentation Servers. However, all Presentation Servers communicate with a single Process Server. Again, the Process Server's HTTP-Connector attributes for *proxyName* and *proxyPort* need to be adjusted, pointing to the front-end load balancer.

2.2.4.2. *Single Presentation Server - Clustered Process Server*

In this case, a Load Balancer is placed between the Presentation Tier and the Application Tier and all Process Servers' HTTP-Connector attributes for *proxyName* and *proxyPort* need to be adjusted, pointing to the Presentation Server.

2.2.4.3. *Clustered Presentation Server, Clustered Process Server*

As in Section 2.2.4.1, a load balancing front-end distributes the load among several Presentation Servers. And like in 2.2.4.2, a Load Balancer is placed between the Presentation Tier and the Application Tier. All Process Servers' HTTP-Connector attributes for *proxyName* and *proxyPort* need to be adjusted, pointing to the load balancing front-end.

2.2.5. *Intra-Tier Communication*

Clustering a component in the Presentation or Application Tier will lead to intra-tier communication via a network.

Cluster nodes broadcast updates about their current status and use TCP Socket communication for exchanging information. This is described in more detail in the next section.

Additionally, clustered Process Server nodes communicate via remote message invocation (RMI). By default, an RMI registry is started on one of the clustered Process Server nodes on port **10990**. All participating

process engines in the cluster act as RMI clients. For this reason, it is recommended that the cluster be set up on an isolated network.

All of the nodes monitor the status of the registry server and in the event of that server failing, one of the existing nodes takes over the role of registry server.

2.4. Application Tier <-> Data Tier

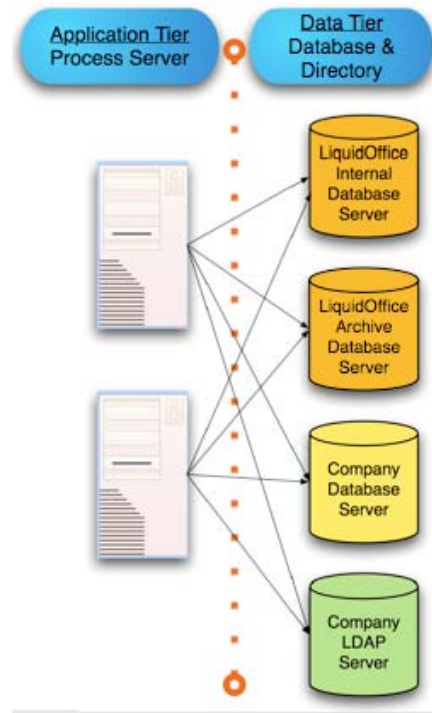


Figure 4: Application Tier <-> Data Tier

2.4.1. LiquidOffice Process Server <-> Database Server(s)

LiquidOffice Server establishes multiple connections to the Database Server(s) using JDBC. The default ports used by the internal database servers are:

- Oracle: **1521**
- Microsoft SQL Server: **1433**

Virtually every request initiated in the Client Tier requires some LiquidOffice Process Server to Internal Database Server communication.

All data, except passwords, are sent in clear text to the Internal Database Server. Since passwords are encrypted on the LiquidOffice server before being stored in the internal database server, they are sent encrypted via the available communication channel.

2.4.2. LiquidOffice Server <-> External Database

When LiquidOffice administrators set up data Connect Agents to external databases for designers to use in their forms, additional communications links may be necessary. Designers may use these Connect Agents to have their form data exported to or to set up database validations, etc.

When form designers choose to use one of these Connect Agents, the LiquidOffice Server will establish connections to the server hosting the external database as necessary. The ports used (if required by the specific database) are those specified for the external database server when communicating via JDBC.

2.4.3. LiquidOffice Server <-> LDAP Server

LiquidOffice Server establishes multiple connections to each LDAP Server. The port used is defined when the LiquidOffice administrator sets up the connection to the LDAP Server in the Management Console. The standard port is 389 for LDAP and 636 for LDAPS.

All of the directory queries are sent over these connections using the LDAP version 3 protocol. If LDAPS is not used, all data is sent in clear text to and from the LDAP Server. If LDAPS is used, all communication between LiquidOffice and the LDAP server is encrypted.

2.4.4. LiquidOffice Server <-> Other

Administrators can write JavaScript that runs on the server. The JavaScript written by administrators can access, directly through the built-in classes, the file system (including network drives) and external databases or the Connect Agents set up through the administration UI. Each of these will require the respective ports to be open.

Form Publishers can also write server script that gets published with the form. For Form Publishers, the built-in class for file access limits them to a relative path only and the built-in class for database access limits them to using Connect Agents for which they have been given permission by a LiquidOffice administrator.

Process Publishers can also write server script that gets published with the process.

Administrators and publishers can access Java classes directly and therefore access other machines and services as required. The ports required will vary depending on the service being used.

3. Clustering

Clustering LiquidOffice can greatly improve availability and scalability of the server. LiquidOffice Clustering sits on top of Tomcat clustering and implements the *Sticky-Session Without Session Sharing* clustering model.

HTTP is a stateless protocol and to keep state (e.g. so that a client doesn't have to send user credentials with every request), a session needs to be maintained on the server. The session is identified by a session-id that the browser received in a cookie after a successful login and returns with every further request.

Session data is not synchronized or shared among cluster nodes, which means that the load balancer has to ensure that a single client's requests are always routed to the same cluster node.

Load balancers usually accomplish this by keeping track of the cookie, which is used to identify the session. LiquidOffice ensures that all issued session-ids are truly cluster-unique.

Sticky-Sessions without Clustered Session Sharing does not require a `<distributable/>` tag in a web applications deployment descriptor. In fact, including it would trigger the unnecessary attempt to synchronize session data among all participating cluster nodes.

3.1. Cluster Tag

LiquidOffice clustering is based on Tomcat's implicit clustering capabilities, which are enabled by activating the `<cluster>` section in Tomcat's server configuration file (`($CATALINA_BASE/conf/server.xml)`), which loads `org.apache.catalina.cluster` package.

```
<Cluster className="org.apache.catalina.cluster.tcp.SimpleTcpCluster"
expireSessionsOnShutdown="false"
managerClassName="org.apache.catalina.cluster.session.DeltaManager" name="Liquid
Office Cluster" useDirtyFlag="true" replicationMode="pooled" >

<Membership className="org.apache.catalina.cluster.mcast.McastService"
mcastAddr="228.0.0.4" mcastDropTime="3000" mcastFrequency="501" mcastPort="45564"
/>

<Receiver className="org.apache.catalina.cluster.tcp.ReplicationListener"
tcpListenAddress="auto" tcpListenPort="4001" tcpSelectorTimeout="100"
tcpThreadCount="6" />

<Sender className="org.apache.catalina.cluster.tcp.ReplicationTransmitter"
replicationMode="pooled" />
<Valve className="org.apache.catalina.cluster.tcp.ReplicationValve"
filter=".*\.(gif|.*\.(js|.*\.(jpg|.*\.(htm|.*\.(html|.*\.(txt|
<Deployer className="org.apache.catalina.cluster.deploy.FarmWarDeployer"
deployDir="/tmp/war-deploy/" tempDir="/tmp/war-temp/" watchDir="/tmp/war-listen/"
watchEnabled="false" />
```

```
</Cluster>
```

Attribute values in the server descriptor (server.xml) are set based on choices during install but can be fine-tuned by directly editing the file. The cluster tag inside the server.xml configurator, defines which ports the LiquidOffice cluster nodes use for broadcasting and peer-to-peer communication.

When a node starts, and frequently for as long as it remains running, it broadcasts a small token identifying itself and its current state to other cluster members. The broadcast address defaults to **228.0.0.4** and the multicast port to **45564**.

The multicast port needs to be set during install and using the default port could have the effect that other Tomcat cluster nodes, although not having LiquidOffice installed, would still try to join the LiquidOffice cluster. Therefore, it is not recommended to use the default port. Instead use any other 5-digit number that is not used to multicast cluster communication on your network.

Broadcasting is used to inform other nodes about the sender's state (e.g. "starting up", "still alive", "shutting down") but not for exchanging information between nodes.

Inter-node communication happens through TCP/IP sockets. By default, LiquidOffice Cluster nodes listen on port **4001** for communication requests from their peers. Additional attributes allow for fine-tuning cluster behavior:

- `Cluster.expireSessionsOnShutdown`: Indicates if sessions should be expired upon shutdown of the node.
- `Receiver.TcpThreadCount`: The number of TCP threads used to handle incoming requests from peers.

3.2. Front loaded Load Balancing

In a scenario where the name (or IP Address) of the load balancer is used to address the LiquidOffice Server, Tomcat needs to be made aware of this name (or IP Address) and respond appropriately. (In our model, this would place the Load Balancer between the client tier and the presentation tier). This is done by setting the *proxyName* and *proxyPort* attributes in the Connector tag, inside the Server.xml configuration file.

```
<?xml version="1.0" encoding="UTF-8"?>

<Server port="8005" shutdown="SHUTDOWN" debug="0">

..

<Service name="Catalina">
<!-- Define a non-SSL Coyote HTTP/1.1 Connector on port 80 -->
<Connector port="80" maxThreads="150" minSpareThreads="25" maxSpareThreads="75"
enableLookups="false" redirectPort="8443" acceptCount="100" debug="0"
connectionTimeout="20000" disableUploadTimeout="true" proxyName="
Balancer.company.com" proxyPort="80" />
..
```

```

</Service>
</Server>

```

The values for the *proxyName* and *proxyPort* attributes are set automatically, based on choices during install but can be fine-tuned by directly editing the file.

The <Connector> element is configured as a child of the <Service> element and cannot be configured as a parent to any element.

3.3. Clustered Process Server Files

In a clustered environment, most of the resources are stored on a shared network drive, accessible by all nodes. Files and Folders that are kept on each node are listed here:

Windows:

Folder / File	Comment
Program Files/Cardiff/LiquidOffice/xmlserver/jre	Java Runtime Environment
Program Files/Cardiff/LiquidOffice/xmlserver/tc	J2EE Servlet Container
Program Files/Cardiff/LiquidOffice/xmlserver/tc/server/lib	LiquidOffice Cluster Implementation library: locluster.jar
Program Files/Cardiff/LiquidOffice/xmlserver/tc/common/lib	LiquidOffice Cluster Implementation library: loclustermessage.jar
Program Files/Cardiff/LiquidOffice/xmlserver/xmlbase	This is the CATALINA_BASE directory
Program Files/Cardiff/LiquidOffice/xmlserver/xmlbase/conf	Nodes' Servlet Container configuration files
Program Files/Cardiff/LiquidOffice/xmlserver/xmlbase/conf/Catalina/localhost	Node's Context descriptors

Solaris

Folder / File	Comment
/usr/local/cardiff/liquidoffice/xmlserver/jre	Java Runtime Environment
/usr/local/cardiff/liquidoffice/xmlserver/tc	J2EE Servlet Container
/usr/local/cardiff/liquidoffice/xmlserver/tc/server/lib	LiquidOffice Cluster Implementation library: locluster.jar
/usr/local/cardiff/liquidoffice/xmlserver/tc/common/lib	LiquidOffice Cluster Implementation library: loclustermessage.jar
/usr/local/cardiff/liquidoffice/xmlserver/xmlbase	This is the CATALINA_BASE directory

/usr/local/cardiff/liquidoffice/xmlserver/xmlbase/conf	Nodes' Servlet Container configuration files
/usr/local/cardiff/liquidoffice/xmlserver/xmlbase/conf/Catalina/localhost	Node's Context descriptors

Linux

Folder / File	Comment
/usr/local/cardiff/liquidoffice/xmlserver/jre	Java Runtime Environment
/usr/local/cardiff/liquidoffice/xmlserver/tc	J2EE Servlet Container
/usr/local/cardiff/liquidoffice/xmlserver/tc/server/lib	LiquidOffice Cluster Implementation library: locluster.jar
/usr/local/cardiff/liquidoffice/xmlserver/tc/common/lib	LiquidOffice Cluster Implementation library: loclustermessage.jar
/usr/local/cardiff/liquidoffice/xmlserver/xmlbase	This is the CATALINA_BASE directory
/usr/local/cardiff/liquidoffice/xmlserver/xmlbase/conf	Nodes' Servlet Container configuration files
/usr/local/cardiff/liquidoffice/xmlserver/xmlbase/conf/Catalina/localhost	Node's Context descriptors

4. Configurations

There are many possible configurations of LiquidOffice. The specifics of an organization's network topography will have the largest impact on what configuration is eventually chosen by the organization.

Because communication cannot skip over a tier, it is possible to isolate components in each tier, e.g. setting up servers with two network cards, one network card communicates with each connected tier.

4.1. Default Ports

Tier	Client Port	Server Port	Usage	Comment
Client	80	N/A	HTTP	
Client	443	N/A	HTTPS	optional
Web Server	80	80	HTTP	
Web Server		443	HTTPS	optional
Web Server	4001	4001	TCP	only if clustered
Web Server	mcastPort	228.0.0.4	Broadcast	only if clustered
Application		80	HTTP	
Application	1433		Database	MS SQL Server
Application	1521		Database	Oracle
Application	10990	10990	RMI	only if clustered
Application	389		LDAP	Optional
Application	636		LDAPS	Optional
Application	mcastPort	228.0.0.4	Broadcast	only if clustered
Data		1433	Database	MS SQL Server
Data		1521	Database	Oracle
Data		389	LDAP	optional
Data		636	LDAPS	optional

