

# Brio.Portal Planning

The top issues to consider when planning for an enterprise information portal are to build a system that:

- Can support your users and their concurrent requests. Regardless of whether one user or several thousand users are accessing Brio.Portal at the same time, each one needs to perform basic activities in a timely manner. These activities include logging in, browsing, listing files, downloading files, searching, refreshing, publishing, and running jobs. Users must consistently be able to perform these activities using portal software as a single point of access to company information.
- Delivers scalability for the expanding enterprise. Because Brio.Portal has a multitier, distributed architecture, it can scale to meet the challenges that typically face today's enterprises:
  - Supporting large numbers of geographically distributed concurrent users (logged-in and use Brio.Portal at the same time).
  - Supporting the stored objects, information searches, and concurrent jobs required by a worldwide user base.
  - Creating content from a multitude of diverse distributed data sources.
- Is easy to support and maintain.

## Brio.Portal's Scalable Architecture

Brio.Portal has a multitier, Web-based distributed architecture. This architecture enables data access from disparate enterprise systems. Staging the data as files, reports, and other program output in a middle tier repository, Brio.Report then delivers requested documents to the end-user through a Web browser. Using multiple distributed components, information is accessed from enterprise systems, condensed into reports, and delivered to any user with a Web browser.

Brio.Portal's distributed components essentially fall into three logical tiers:

- **Client management tier.** Accepts client requests and transfers these to the session management tier. Accepts results and presents them to the client.
- **Session management tier.** Manages client requests, routing them to the appropriate back-end service.
- **Back-end service tier.** Acts on client requests and returns the results through the session management tier.

Ultimately, Brio.Portal's multitier, distributed architecture results in the following benefits:

- **Improved performance for increasing numbers of users and increasing Brio.Portal content.** Brio.Portal shifts costly session management to the middle tier, improving performance for increasing numbers of users. By replicating and distributing Brio.Portal services, you can sustain high performance as the content in the Brio.Portal system grows.
- **Greater flexibility for scaling enterprise infrastructures.** Brio.Portal's middle tier services, which separate clients from back-end services, allow information management to transparently grow and change the system without affecting users.
- **True reliability, availability, and serviceability.** Brio.Portal enables replication of services to execute client requests and allows load balancing among identical services to ensure availability. When system management needs to scale Brio.Portal, it can add components to the system without disrupting operation or affecting clients.
- **Increased security.** Because Brio.Portal handles session management in the middle tier, clients do not have direct access to back-end services. Instead, these middle tier services function like an application firewall that protects critical business intelligence.

## **Brio.Portal Services**

In Brio.Portal, several cooperative services store, manage and deliver information objects. These services each handle a specific activity and are designed to ensure a scalable, broad-based reporting and information delivery system.

A servlet presents the user with a graphical interface through which to browse the Brio.Portal repository, view and download documents, check notifications, and generate customized reports. A servlet is a process that executes on a Web server.

Brio.Portal provides a graphical tool for administering the system. This tool enables the administrator to perform the tasks of managing user accounts, access to repository objects, monitoring the operation of the Brio.Portal services, and scheduling system-wide events.

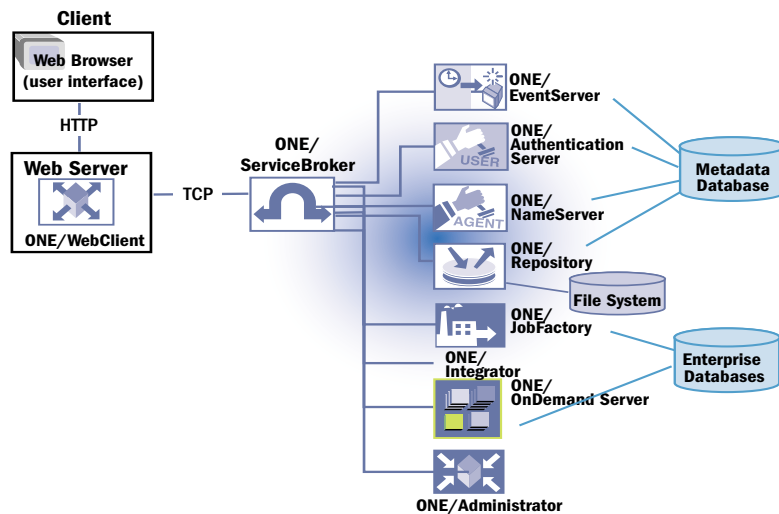


Figure 1 Brio.Portal Cooperative Services

## ONE/WebClient

Users connect to the Brio.Portal system through ONE/WebClient. Each user's view of Brio.Portal is customizable, according to user permissions and personal preference. One user might receive updates to reports, another click on a particular internet bookmark, and yet another perform interactive data analysis, and so on.

## Look and Feel

You can customize ONE/WebClient to integrate with your company's current network system and its "look and feel."

Using ONE/WebClient configuration tool, you can modify such conventions as how your users log in, what the first screen displays, or in what language the system communicates (locale). Brio.Portal supports multiple language/character sets for geographically distributed office.

You can customize the form that collects information for a program that will generate a report. You can also customize the user interface either by modifying ONE/WebClient templates or by using ONE/SmartCuts to display customized web pages. For more information, refer to *ABCs of Customizing the Brio.Portal UI* (tentative title of the paper), which will be available in April.

## What ONE/WebClient Does

ONE/WebClient accepts and reviews end-user requests, sending off the work for servicing, and presenting the results in the form of HTML pages and HTTP replies. ONE/WebClient passes all searches, retrievals, and job execution to the Brio.Portal system through ONE/ServiceBroker.

## ONE/ServiceBroker

All Brio.Portal services connect to the system through ONE/ServiceBroker, which performs session management, service request routing, dynamic load balancing, and security management.

### Session Management

The primary role of ONE/ServiceBroker is to provide overall session management for Brio.Portal. ONE/ServiceBroker sets up and maintains each end-user connection, routes client requests to appropriate services, dynamically load balances between different services, and enhances security.

Because the end-user is never aware of any service agent's location, the administrator can easily move a service agent to a new machine or perform other administrative tasks, without any end-user impact.

### Service Request Routing

ONE/ServiceBroker routes client service requests to the appropriate Brio.Portal service agent. The requesting client does not have to know the location or function of any Brio.Portal service agent. This allows the Brio.Portal administrator flexibility in transparently locating and moving service agents.

### Dynamic Load Balancing

ONE/ServiceBroker distributes work among Brio.Portal service agents that support identical services. For example, an administrator can configure two ONE/JobFactories to run the same type of reports. When two ONE/JobFactories provide replicated services, then ONE/ServiceBroker will dispatch work in balanced amounts between the two ONE/JobFactories.

This load balancing improves performance since two machines can process ONE/JobFactory requests in parallel. Another advantage of providing replication is to support fault tolerance. If two different ONE/JobFactories provide identical services and one ONE/JobFactory is down, then the Brio.Portal system would continue to operate properly as ONE/ServiceBroker dispatches requests only to currently operational ONE/JobFactories.

### Security

ONE/ServiceBroker serves as an application firewall. Clients cannot connect directly to a Brio.Portal service agent, but must pass through ONE/ServiceBroker, which then routes the request to the appropriate service agent.

## ONE/NameServer

Brio.Portal services, or *service agents*, obtain their startup information from ONE/NameServer, which manages their configuration information. ONE/NameServer provides a directory lookup service to all the other Brio.Portal service agents.

## Maintaining Service Agent Information

Each service agent requires this information when it starts up. At service agent startup, ONE/NameServer authenticates each service agent by name and password and the port number at which the service agent will “listen.” ONE/NameServer keeps the information in a configuration file. In addition, this configuration contains additional information that some services require, such as the name and location of ONE/Repository RDBMS and the directory to which a service agent writes log information.

## ONE/AuthenticationServer

The service agent that authenticates users who connect to ONE/WebClient is ONE/AuthenticationServer. It lets users connect either through the Brio.Portal native authentication driver or through your company’s own authentication system.

## Authentication Server Repository

ONE/AuthenticationServers maintain user and group properties in the Authentication Server Repository (ASR), storing the ASR in an RDBMS in a way similar to that used by ONE/NameServer. However, the databases of the two service agents are distinct from each other.

## ONE/Repository

Each Brio.Portal installation maintains one and only one ONE/Repository, which manages the storage of objects. ONE/Repository provides central storage for reports and report output, queries and their results, as well as related objects, such as HTML files, graphics, and executables.

Only ONE/Repository has direct access to storage. Other agents access objects by connecting to ONE/Repository through the network. ONE/Repository processes requests to store, retrieve, search, and browse and returns a result set to the requestor.

## Storing Objects

ONE/Repository maintains an information object catalogue in a relational database management system (RDBMS) outside of the Brio.Portal system.

ONE/Repository keeps *metadata* about these objects, including MIME-type information, access-control permissions, description, keywords, version, expiration date, category hierarchy, and relationships between objects. This metadata allows Brio.Portal to relate reports and queries to their outputs, images to HTML files that include them, and so on. ONE/Repository separately maintains the objects in its file system, in the subdirectory called *server*.

Brio.Portal supports the following databases: Microsoft SQL Server, DB2, Informix7, Oracle7, Oracle8, and Sybase.

## Displaying Repository Contents

ONE/Repository uses a tree-like structure of user-defined content categories. The category structure serves as the Brio.Portal navigational guide. Starting with a root category, the user can navigate down categories and subcategories.

There are no internal Brio.Portal limitations on the number of subcategories that can exist within a given category nor are there any constraints on the number of objects that can be placed within a category or sub-category.

## ONE/EventServer

ONE/EventServer provides scheduling, subscription, and event notification services. It dispatches scheduled jobs for execution by One/JobFactories, enables users to subscribe to and receive notifications, and notifies users when relevant events occur.

## Sending Notifications

ONE/EventServer sends notifications by email, sending these out on a periodic basis. Optionally, ONE/EventServer can also send a file attachment.

## ONE/JobFactory

From ONE/WebClient, a user enters a Brio.Portal job request to run a program, such as SQR or Brio Reports, and gets back the report output. ONE/JobFactory manages execution of this job. ONE/JobFactory accepts job requests, manages job execution, and returns results both to the end-user through ONE/WebClient and to the repository for future output distribution.

## Job Execution

According to job requirements, ONE/JobFactory sends the job to the appropriate applications, such as Brio's SQR, Oracle's SQL\*Plus, Sybase's Transact SQL, or a batch scripting engine such as PERL. ONE/JobFactory controls the flow into and out of the underlying application engine, creating separate operating system processes for each job request.

## ONE/OnDemand Server

ONE/OnDemand Server (ONE/ODS) is a request server that manages requests and delivers BrioQuery report folios to Web browsers. ONE/ODS allows the user to:

- View and select documents from a list
- Modify and re-process queries

Configuring ONE/ODS as a Brio.Portal component facilitates centralized administration of users, groups, permissions, and user access to documents stored in the repository. This centralized administration allows control of the following:

- Which documents a specific user can see
- What permissions the user has for a particular document

---

**Note** When installing Brio OnDemand Server as a Brio.Portal service agent, you must also install ONE/Integrator and ONE/Administrator on the same machine as ONE/ODS.

---

## ONE/Integrator

Installing ONE/Integrator, which is on the Brio.Portal CD under separate licensing, makes available a collection of Java APIs and sample programs designed to access Portal features and capabilities from other programs. For example, ONE/ODS integrates with Brio.Portal using APIs made available by ONE/Integrator. Its primary use to organizations has been to write Java utilities to “bulk load” objects into Brio.Portal.

## ONE/Administrator

The administrator's primary tool is ONE/Administrator. It has a set of “wizards” that simplify management of Brio.Portal users, objects and categories, services, and schedules.

Routine activities that ONE/Administrator facilitates are adding, modifying, deleting of users, categories, and scheduled events. It also assists the administrator in more complex tasks of managing groups, categories, and jobs for more efficient and secure use of Brio.Portal.

## Planning Your Initial Configuration

Brio.Portal software components, or service agents, work cooperatively to create an efficient and easy-to-use information reporting and delivery system. Having discussed the function of each service agent and how the service agents work together, this section suggests an approach to deploying your initial Brio.Portal configuration.

How will you use Brio.Portal at your company? As you plan your Brio.Portal implementation consider the top performance factors for Brio.Portal systems: user activity and taxonomy (how grouped) and content management. Determine the resource requirements for Brio.Portal components with these factors in mind.

To develop the appropriate framework for your Brio.Portal, consider the following questions:

- Who will the users be?
- What are the existing organizational structures?
- Do any of these structures reflect the relationship between users and data storage and access?

## Planning User Accounts and Authentication Schemes

As you plan how to organize users, determine whether you want to build a new list of users or adopt an existing user profile for Brio.Portal. Consider the following questions as you plan the Brio.Portal user structure and the way you want to authenticate users. Refer to the Security white paper for more information regarding security issues raised in this section.

- Will users log into Brio.Portal once at the beginning of the day or several times during day? The log-in pattern that you anticipate will determine how the administrator sets the Brio.Portal time-out session.
- Does your company prefer one login procedure that will transparently log in users to Brio.Portal when they log into the company intranet? This determines whether you configure an external authentication server.
- Will you be using an external directory for user authentication?

If so, you will need to decide on whether to use a Lightweight Directory Access Protocol (LDAP)-based or NT-based system or an internally developed authentication system.

- Will your Brio.Portal have external users that do not work for the company, such as customers or clients.

You will not be able to always assume that the external user has anything more than a Browser on the PC. Nor can you assume that they have a certain version of a Browser. The following are determinations that the customer will have to make.

- Whether to have these users to dial-in from behind a firewall or use a virtual private network (VPN).
- Whether to provide these users with a URL to an Internet site hosted by your company's Brio Portal, that will allow them direct access to Brio.Portal?
- Are there any web-based authentication applications currently in use in your organization? This would be a front-end system that authenticates the user for access to the Web server and once authenticated, allows user access to the applications running on the Web server.
- What is your security infrastructure? Security becomes particularly important when you make some data accessible to customers. Are there firewalls? Firewalls are computers or communications devices that filter access to a protected network. They protect central resources from general access and preserve confidentiality of information as it passes through an intranet or extranet.
  - Where are the Web servers in relation to the firewall (inside or outside)?
  - What Web servers are in use by your organization (IIS, Netscape, Apache, other)?

## Identifying Users

There probably are very few instances that require building a user list from scratch. If your company has user profiles currently in use, see whether you can build on one of these user profiles. It might actually be a requirement that you use an existing user profile. If you are developing a new user list, list them according to the following activities that Brio.Portal users perform:

- Search, view, and download documents, reports, or other objects
- Run jobs (such as financial reports or sales reports)
- Publish reports into the Brio.Portal repository
- Support and maintain the system
- Develop applications or customize interfaces to give the Brio.Portal system the “look and feel” of your company intranet

## Grouping Users

Brio.Portal facilitates access to data by assigning users to groups. Groups that already exist within your company might suggest a pattern of access that you can use for Brio.Portal. Some examples of existing groups:

- Business units
- Geographical locations
- Projects
- Enterprise resource planning (ERP) hierarchical trees
- Departments: human resources, marketing, sales, engineering, customer services, and so on
- People with the same job title
- Structure defined in the external authentication directory

In a few cases, reports might be associated with individuals who publish them.

You can form your groups according to an existing organizational structure. For example, you might use project names as group names, because project participants will be the principal users of project data. If access will be by locations then group users by location: for example, Paris, London, New York. Or you might derive group names from two existing structures, such as department names and project names. A user can belong to multiple groups, but must start with one default group.

Identify external systems where group organization exists and determine if one of these systems meets your requirements for grouping within Brio.Portal.

## Planning for Efficient Category Structures

Enterprises typically use Brio.Portal to store massive amounts of information, organized into logical categories that users browse.

## Content

Before you can plan your category structure, you must plan the types of content that you hope to deploy initially. For example, your organization might want to distribute reports from a specific application, such as PeopleSoft HRMS or Oracle Financials. Certain users might need the reports in certain types of formats, for example, as Excel spreadsheets or HTML files.

Consider next how you want to load this content into Brio.Portal. You can load content automatically from an existing system, or manually. In addition to static report output, you can enable on-demand report or job execution. Determine which reporting applications you will make available within Brio.Portal.

Next, plan for the number of objects you will initially store in Brio.Portal. These objects include files (such as report output files, HTML files, spreadsheets, and so on), reports (including all files needed by each report at runtime), and names and types of applications needed for execution or loading of reports and files.

## Search Path

Plan Brio.Portal categories that replicate the navigational interface on internet Web sites. For example, to enable a search flow from general to specific, your Brio.Portal search path through categories might be “Company Home > Sales Division> Western Region > California”. Keep this navigational flow in mind when planning each user’s default category. The user’s default directory is the user’s “home” in Brio.Portal, where Brio.Portal stores output of reports run and documents downloaded by the user. Select a category that would be the logical home of the user within Brio.Portal.

## Objects in a Category

Create a directory structure that limits each category to less than 200 objects. The time Brio.Portal takes to browse or refresh a category is proportional to the number of objects being retrieved. Since Brio.Portal enables you to create highly complex category structures, take advantage of this flexibility to improve performance. Although ONE/WebClient will display any number of objects in a category, keep fewer than 200 objects in any one category whenever possible.

## User Default Category

Create a user’s default category in the search path where you expect her to do the most browsing. For example, given the ... *Sales Division > Western Region > California* search path example, you would assign a default category for a sales person in California in a subcategory of *California*, rather than to a search path that is derived from the *Engineering* path.

## User Permissions

Set up categories according to user permissions. Categories that contain only those objects that groups of users can access makes efficient use of Brio.Portal. Categories that contain many objects to which groups of users do not have access is inefficient. This is because every time a user drills into a category, Brio.Portal has to search through all of the objects in that category to display the objects to which a user has access, whether these be 5 of 200, or all of 200 objects.

Place reports for which all users have read access in one category. Place confidential reports available only to select group (for example, executive staff) in a separate subcategory. Adding a few generally-accessible reports to a category where hundreds of reports are confidential is inefficient. When a “ordinary” user browses such a category, Brio.Portal will have to search

through all the reports in order to display the few that are generally viewable. When you remove the few generally-accessible reports from such a “confidential” category, then the category will not be visible to the ordinary user.

### Speeding Up Information Access

Besides the way you plan your category structure and where you place users’ default categories, you can speed up information access by taking these performance-enhancing measures:

- **Set up jobs so that Brio.Portal automatically deletes the output within an appropriate time frame.** Brio.Portal offers this expiration feature so that you can ensure your system does not become cluttered with old files.

---

**Note** Show your users how to download, rename the output, and save it on their own machines.

---

- **Run Brio.Portal on a dedicated, distributed system.** As explained in the next section, there are many ways to distribute Brio.Portal services so that your system can grow to meet enterprise demands.
- **Implement general database and O/S tuning procedures to ensure optimal operation of Brio.Portal.** This is especially important in systems where customers run many reporting jobs, where disk writes might be performed on both the database to which Brio.Portal connects for executing the job and the disk on the server with ONE/Repository.

### Estimating Brio.Portal Resources to Support User Activity

When planning the hardware required for Brio.Portal, you need to estimate how many users will be logging in to the network and how they will use Brio.Portal. Determine how many users will be accessing Brio.Portal at the same time (concurrent users). From existing patterns of activity, such as intranet use, estimate the login pattern of your users. Estimate how many reports the Brio.Portal users will run and how long the reports will be. Consider the questions in Table 1 to help you estimate total user activity.

**Table 1** Estimating Total User Activity

---

How many concurrently connected users do you anticipate?
Are users geographically distributed?
How many users will log in per minute?
How many reports will run concurrently?
How many pages of output will generated reports contain?
How many objects do you anticipate storing in the Brio.Portal repository?

---

Use the answers to these questions against the values shown in the tables that follow. The tables show Brio.Portal service agents CPU and memory requirements and Oracle tablespace requirements for the following platforms:

- Solaris
- Windows NT

---

**Note** The amount of disk (file system) space needed for Brio.Portal depends on the number and size of objects (documents and other files) to be stored in Brio.Portal. Determine how much disk space those objects require, allowing extra space for objects to be added later and for output from reporting jobs, then add 50 MB of space for the Brio.Portal software and demo data.

---

### Estimated System Resource Guidelines: Solaris

Table 2 and Table 3 guidelines for estimating Solaris Brio.Portal resource requirements are based on the following test configuration. Table 4 shows Oracle 8.0.4 tablespace requirements.

- Solaris Server with Brio.Portal agents and ONE/WebClient:
  - **Operating System:** Solaris 7
  - **Processors:** 2 – 296 MHz Sparc processors:
  - **Memory:** 1GB
  - **Disk:** Single 9.1 GB, 7200 RPM EIDE drive, 16.7 MB/sec internal disk I/O
  - Sun JRE 1.1.7
- Database server used with the test system:
  - **Database:** Oracle 8.0.4
  - **Operating System:** Windows NT 4.0
  - **Processors:** 2 - 400 MHz Pentium II processors
  - **Memory:** 1 GB
  - **Disk:** RAID 5 striped over 3 - 9.1 GB drives

All Brio.Portal service agents were installed on a Solaris server and the database was installed on a Windows NT server. Table 2 provides guidelines for estimating system resource requirements based Brio.Portal components and the RDBMS for the Brio.Portal metadata. To estimate resource requirements for additional users and/or SQR jobs, consult Table 3.

**Table 2** Estimated Solaris System Resource Guidelines

(Based on: ≤100 concurrent users; ≤1 SQR job at a time; ≤10 pages/job output)

<b>Brio.Portal/Other Software</b>	<b>CPU</b> (fraction of)	<b>Memory</b>	<b>Heap Size</b>
Name Server	0.1	40 MB	Default
Repository	0.5	100 MB	20 - 80 MB
Event Server	0.1	15 MB	Default
Service Broker	0.3	50 MB	20 - 60 MB
SQR Job Factory	0.1	30 MB	Default
Authentication Server	0.2	30 MB	Default
WebClient	2.0	75 MB	20 - 75 MB
Oracle (Brio.Portal)	0.1	100MB	N/A

**Note** Brio.Portal CPU requirements are based on 296 MHz SPARC Ultra-Ili processors. Oracle CPU requirements are based on 400 MHz Intel Pentium II processors. The unit used for CPU requirements is *fraction of CPU* in order to facilitate calculation of number of CPU required for the system.

**Table 3** Estimated Solaris System Resource Scaling Factors for Brio.Portal Agents Only

	<b>For 100 Additional Concurrent Users, Add:</b>		<b>For each Additional Concurrent SQR Job, Add:</b>	
	CPU (fraction of)	Memory	CPU	Memory
Name Server	0.01	--	--	--
Repository*	0.5	30 MB	0.03	20 KB
Event Server	--	--	(No Data)	
Service Broker	0.25	15 MB	0.01	75 KB
SQR Job Factory	--	--	0.1	5-10 MB
Authentication Server	0.04	1 MB	--	--
WebClient	2.0	15 MB	--	--
Oracle (Brio.Portal)	0.10	50 MB	0.02	500 KB

\*For categories containing more than 50 objects, add 120 KB of memory per object for the Repository.

---

**Note** Brio.Portal CPU requirements are based on 296 MHz SPARC Ultra-Ili processors. Oracle CPU requirements are based on 400 MHz Intel Pentium II processors.  
ODS, which was not part of the test configuration, requires a minimum of 60 MB of hard disk space and 64 MB of RAM.

---

**Caution!** When using the information in Table 3 to scale to 2,000 concurrent users (who pause to think), plan for an additional 10-20% capacity.

The tablespace requirements shown in Table 4 are for Oracle 8.0.4 installed on a Windows NT system.

**Table 4 Oracle 8.0.4 Tablespace Requirements**

Action	Tablespace Used
Install empty Brio.Portal with demo data	3 MB
Add 1000 more users	0.5 MB
Add 1000 more groups	0.1 MB
Add 1000 more categories	3.0 MB
Add 1000 more objects (HTML files)	2.5 MB
Add 1000 more SQR jobs	4.0 MB

### Resource Guidelines for Windows NT-based Systems

Table 5 and Table 6 guidelines for estimating NT Brio.Portal resource requirements are based on the following test configuration:

- All Brio.Portal service agents and ONE/WebClient on one machine:
  - **OS:** Windows NT 4.0
  - **Processors:** 4 - 550 MHz Intel Pentium III processors
  - **Memory:** 1 GB
  - **Disk:** RAID 5 striped across 4 - 9.1 GB drives
  - Microsoft JView Version 5.00.3188
- Database: Oracle 8.0.4
  - **OS:** Windows NT 4.0
  - **Processor:** 2 - 700 MHz Pentium III processors
  - **Memory:** 1 GB

- **Disk:** RAID 5 striped across 3 - 9.1 GB drives  
(One SQR test used RAID 0 striped across 3 - 9.1 GB drives)

**Table 5** Estimated NT System Resource Guidelines

(Based on: ≤100 concurrent users; ≤1 SQR job at a time; ≤10 pages/job output)

<b>Brio.Portal/Other Software</b>	<b>CPU</b> (fraction of)	<b>Memory</b>
Name Server	0.1	40 MB
Repository	0.3	100 MB
Event Server	0.1	15 MB
Service Broker	0.2	50 MB
SQR Job Factory	0.1	30 MB
Authentication Server	0.1	30 MB
WebClient	2.7	75 MB
Oracle (Brio.Portal)	0.1	100 MB

**Note** Brio.Portal CPU requirements are based on 550 MHz Intel Pentium III processors. Oracle CPU requirements are based on 700 MHz Intel Pentium II processors. ODS, which was not part of the test configuration, requires a minimum of 20 MB of hard disk space and 64 MB of RAM.

**Table 6** Estimated NT System Resource Scaling Factors for Brio.Portal Agents Only

	<b>For 100 Additional Concurrent Users, Add:</b>		<b>For each Additional Concurrent SQR Job, Add:</b>	
	CPU (fraction of)	Memory	CPU	Memory
Name Server	0.01	--	--	--
Repository	0.20	30 MB	0.03	40 KB
Event Server	--	--	(No Data)	(No Data)
Service Broker	0.05	15 MB	0.01	130 KB
SQR Job Factory	--	--	0.1	5-10 MB

**Table 6** Estimated NT System Resource Scaling Factors for Brio.Portal Agents Only

	For 100 Additional Concurrent Users, Add:		For each Additional Concurrent SQR Job, Add:	
Authentication Server	0.01	1.5 MB	--	--
WebClient	1.50	10 MB	--	--
Oracle (Brio.Portal)	0.03	50 MB	0.02	500 KB

\*For categories containing more than 50 objects, add 120 KB of memory per object for the Repository.

---

**Note** Brio.Portal CPU requirements are based on 550 MHz Intel Pentium III processors. Oracle CPU requirements are based on 700 MHz Intel Pentium II processors.

---

**Caution** !When using the information in Table 6 to scale to 2,000 concurrent users (who pause to think), plan for an additional 10-20% capacity.

Consult Table 4 for tablespace requirements for Oracle 8.0.4.

## Planning for User Organization

Your Brio.Portal users can consist of a group of employees, some select groups within your organization, or all the employees. In addition, you can grant partners and selected clients access to areas within your Brio.Portal system. Upon identifying who the users will be, you can plan how to organize users into groups for Brio.Portal access. The logical grouping might be by department, geographical, or by business process.

For each user, you can decide which permissions to grant. For some reports, you can specify that any user should be able to browse and print them. For other reports, you can allow users to execute them on demand. You can scale Brio.Portal to enable any number of users to perform any of these activities.

## Configuring Your Brio.Portal Environments

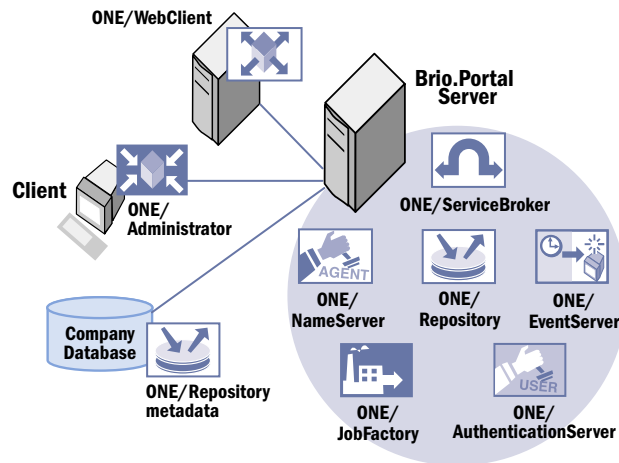
Brio recommends that you implement Brio.Portal in phases, with a test environment to use for proof of concept, a pre-production environment, and finally the production environment. Large installations might also have a separate development (dev) environment to create and test the templates, interfaces, and utilities developed to customize Brio.Portal for the company. Smaller installations might develop customized features and templates in the test environment.

## Test/Dev Environment

The test environment allows you to test implementation and customization, verifying its viability within the total solution.

Guidelines for your Brio.Portal test environment are:

- 1 Work with your database administrator to create a Brio.Portal account and tablespaces for Brio.Portal use in a dedicated database.
- 2 Install Brio.Portal service agents on one server.
- 3 Install ONE/WebClient on the Web server
- 4 Install ONE/Administrator on a client machine.
- 5 Add users, categories, and content that match high priority needs in your company.
  - Use ONE/Administrator if you have a small number of users, objects, or categories.
  - Use the Java utilities in ONE/Integrator to batch-load users, or categories, or objects to Brio.Portal: *LoadUsers*, *LoadCategories*, *LoadFiles*. Each utility processes an input file to load or create the relevant user accounts, categories, and objects. Create the input files using SQR, Excel, or a scripting utility.  
(user: name|pwd|description|defcat|execflag|group|list|defperms, categories:  
path|owner|description|group|perms,  
objects: category|directory|primary)
  - Develop a custom bulk load application using ONE/API.
- 6 If you are integrating Brio.Portal with your company's intranet, extranet, or with other existing systems, familiarize yourself with the information on configuring transparent login (*Administrator's Guide*) and external authentication (Authentication SDK documentation on the Brio.Portal CD) and with the information provided in the security white paper.
- 7 Develop and implement a schedule for backing up the file system on each machine that hosts a Brio.Portal software component and for the repository metadata.



**Figure 2** Sample Brio.Portal Test/Development Configuration

## Pre-Production Environment

The pre-production environment implements your Brio.Portal design. The pre-production environment allows you to install required software, add users and groups, create categories and add content (objects), run jobs against job factories before making the system available to the ordinary user.

### Setting Up the Pre-Production Environment

Having set up your Brio.Portal system platform using the guidelines for estimating system resources, use the following guidelines to configure the Brio.Portal pre-production environment:

- 1** Work with your database administrator to create a Brio.Portal account and tablespaces for Brio.Portal in the database to which Brio.Portal will connect.
- 2** Install SQR Server, other reporting tools, enterprise planning report (ERP) or data warehouse software, as applicable.
- 3** Install Brio.Portal service agents according to your design, following the guidelines and recommendations in this paper.
- 4** Install ONE/WebClient on the Web server or an application server.
- 5** Install ONE/Administrator on a client machine.
- 6** Add users and groups, categories and objects according to your Brio.Portal design.

Migrate users and groups, categories and objects from the test environment (later, from pre-production to production) using the import and export utilities described in the *Brio.Portal Administrator's Guide*. If you used a bulk load utility, use the same utility to load content in your pre-production and production environments.

- 7 Configure ONE/WebClient using the One/WebClient configuration tool.
- 8 Implement the customized features you have developed. Copy over from your test or dev environment the directories that contain the files used in customizing the ONE/WebClient interface.
- 9 Connect to the application servers and/or databases according to your Brio.Portal plan.
- 10 Develop and implement a schedule for backing up the file system on each machine that hosts a Brio.Portal software component and for the repository metadata.

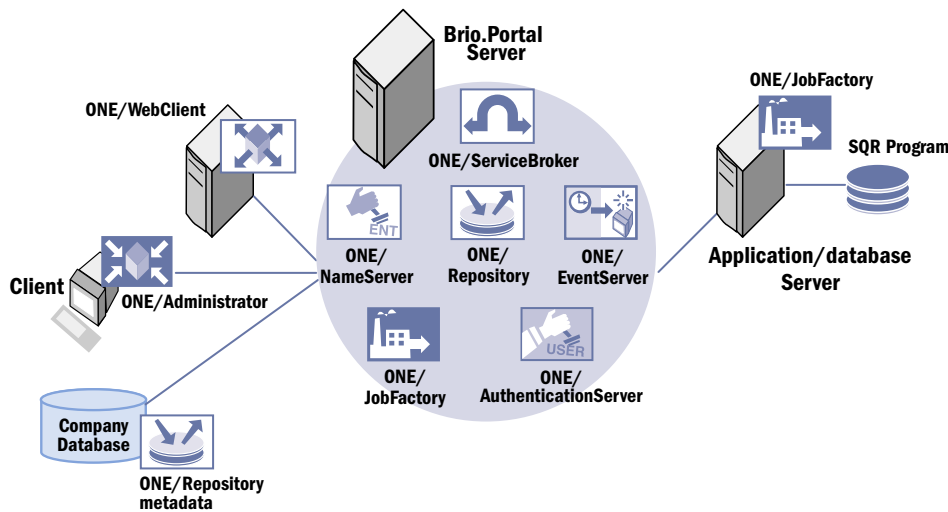


Figure 3 Sample Pre-Production Environment

## Production Environment

The production environment is the pre-production system after you have ensured that everything works as designed.

## Brio.Portal Implementation Example 1

This section walks you through the implementation of Stanford University's Brio.Portal. The Stanford Brio.Portal system provides the infrastructure for creating, storing, running and viewing administrative reports, jobs and files over the network through Brio.Portal.

## Data Sharing and Report Deployment

Stanford's Brio.Portal will serve 1200 named users, with an anticipated 200 concurrent users during busy hours of the work day. Users will generate and publish documents using business applications such as Word and Excel. Multiple projects, each with a dedicated database, will create programs that will be published so that users in these departments can run reports and view report output and other types of objects.

## Environment and Background

The Stanford University installation is campus-wide. One database is currently available for the storing and viewing of files prepared using business applications. Multiple projects, designated as *Application areas*, maintain dedicated databases.

A technical team is responsible for planning, supporting, and growing the Brio.Portal services and operations.

## Implementation Objectives

Stanford University has installed Brio.Portal to meet the following objectives:

- *Share documents* across departments and across campus-wide locations.
- *Create programs* that are published so that users can generate and run reports.
- *Run reports* at scheduled times for users in multiple administrative projects.
- Provide ad hoc querying using BrioQuery and Business Objects and viewing of PDF, Excel, Word, HTML files, and so on.

## Implementation Considerations

Primary considerations for the Brio.Portal installation included the following:

- *Local access:* All users connect to the system from their desktop computers through a local area network.  
  
A central Brio.Portal administrator publishes information to Brio.Portal. Other groups within the company submit information via a drop-off point. The Brio.Portal administrator saves the information and makes it accessible in Brio.Portal.  
  
Stanford plans to eventually allow publishing by departmental administrators.
- *Access from remote sites:* Any users with IP addresses can have dial-up access.
- *Security:* Brio.Portal will limit access to ONE/Repository objects through user permissions. User authentication is integrated with Stanford's implementation of Kerberos (WebAuth). WebAuth authenticates users and passes the information to ONE/AuthenticationServer, so that users enter their login name and password to the Stanford system only once (single sign-on). The ONE/AuthenticationServer determines what permissions the user has within Brio.Portal.

## Brio.Portal Usage Pattern

Users generally connect from their desks to search and download documents and run ad hoc and parameter-driven reports from time to time. The reports vary in complexity and can take from between a minute to an hour to run.

New reports are published into Brio.Portal by the Brio.Portal administrator twice a day.

Table 7 shows the Brio.Portal profile for Stanford University.

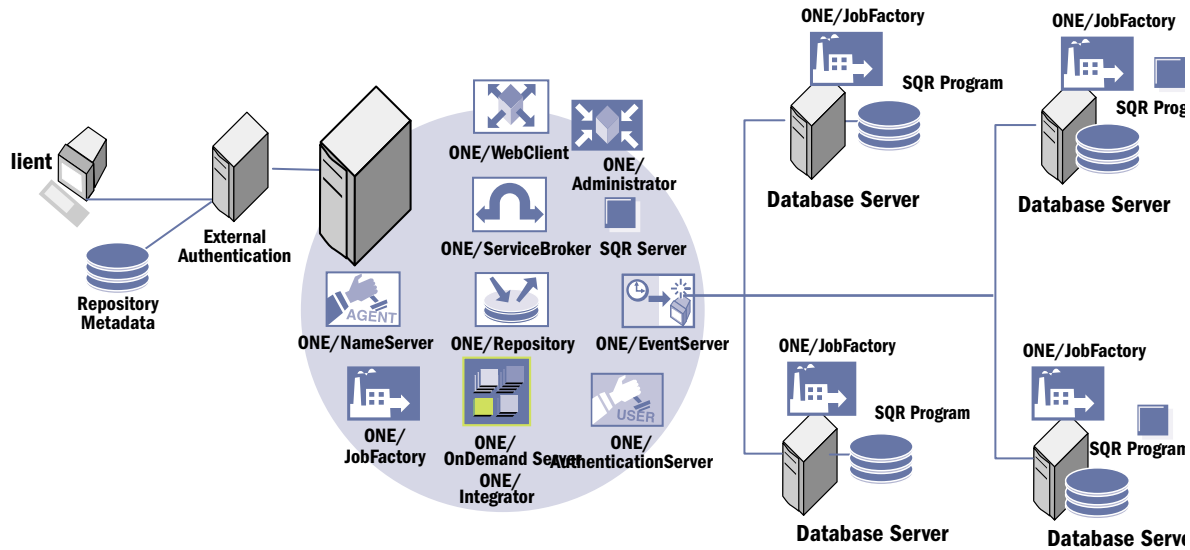
**Table 7** Brio.Portal Profile

<b>User accounts</b>	1200
<b>Concurrent users anticipated</b>	200
<b>Geographical distribution</b>	Campus-wide, some dial-up access
<b>Users logging in per minute</b>	Unknown at this time
<b>Reports to be generated per hour</b>	1 - 5
<b>Pages of output per report</b>	1-20
<b>Estimated number of objects in Brio.Portal</b>	300 - 400 initially

## Deployment

The Brio.Portal components deployed in Stanford's configuration are the following, as shown in Figure 4:

- Brio.Portal service agents on one server: ONE/NameServer, ONE/ServiceBroker, ONE/Repository, ONE/AuthenticationServer, ONE/Job Factory, ONE/ODS, and ONE/WebClient.
- ONE/Repository metadata is stored in an Oracle 8.0.5.
- ONE/Administrator, desktop version, on a client Windows NT machine.
- Databases on separate servers (Oracle, Sybase), for customer care, with one ONE/Job Factory and SQR Server installed on each of these servers. ONE/WebClient runs on an Apache Web server using the Web Logic servlet runner.



**Figure 4** Stanford University Implementation

The Stanford University Brio.Portal provides TCP/IP connection between Brio.Portal and the various RDBMS. User execution of reports, as well as browsing and retrieval of objects is done through the ONE/WebClient interface, and transactions between Brio.Portal and the RDBMS are transparent to the user.

- SQR Server for generating and publishing reports.
- ONE/ODS for BrioQuery reports.

### Pre-Installation Requirements

Prior to Brio.Portal installation, the project database administrator set up a Brio.Portal account in the database and created tablespaces for the persistent storage that Brio.Portal requires. ONE/Repository requires a database to store Brio.Portal metadata and ONE/Name Server and ONE/AuthenticationServer use the same repository for service agent and user account information.

Before installation, the Brio.Portal administrator consulted with various groups to develop a plan for assigning users to default groups. They also utilized a category structure from another campus Web application for Brio.Portal objects.

### Brio.Portal Persistent Storage

Brio.Portal requires a database to store its persistent data. This persistent data consists of Brio.Portal service agent information, end user account information, and the reports and related objects published in Brio.Portal, together with metadata for these objects. A database administrator created the Brio.Portal database and a user account for Brio.Portal.

## Access to Project Data

Access to project data is provided via SQR. SQR is installed on each database server together with a ONE/JobFactory to run reports directly against the production database.

## Categories, Groups, and Users

Stanford University's Brio.Portal category and group structure is identical to that of SUPAD, a Web application that the university uses. SUPAD allows the administrator to add links to other Web sites and links. The SUPAD categories that Brio.Portal has adopted are Computing, Human Resources, Finance, University Policies, and so on.

There are twelve main subcategories under the root. Below these, the number of subcategories will depend on the structure of the content or the need for permissions. Currently, the following is the deepest structure:

- Stanford ReportMart
  - Finances
    - Core Financials
    - Decision Support

Links to other application areas are available within some categories. For example, Decision Support provides a link to the Business Objects WebIntelligence application. In addition, each application area has a link to the application web site for documentation.

To control access to objects in these categories, the Brio.Portal groups created have the same names as the categories. Each user has membership to at least one group, a default group with which objects that the user creates are associated. If the user belongs to multiple groups, then he can choose the group with which to associate objects to be created. This association between object and group facilitates the granting of access permissions. When the user grants *execute* permissions to the group for a report, each member of the group can execute, or run, the report.

User account names are those currently used on the Stanford directory. These do not follow any specific convention. Any changes to userid or password is done in the Stanford directory, not in Brio.Portal. Default groups are assigned according to application area. For example, Core Financials users will be grouped together and will have access to the Core Financials subcategories. Core Financials is a subcategory of the Finance category.

## Brio.Portal Implementation Example 2

This section walks you through a sample implementation with 30,000 users. The Brio.Portal system presented allows users in multiple sites, six sites in this example, to connect to the same database and a server that on which reporting applications and the Brio.Portal ONE/JobFactory reside. We will call this representative example *Bank2000*.

Users access the Bank2000 Brio.Portal from 6 different sites, with each site configured for a different native language (English, French, German, Italian, Belgian, Swedish). The Bank2000 Brio.Portal provides the infrastructure for creating, storing, running and viewing reports that

track topics for users within each site. In addition information across sites is available to selected users, including publishers who make relevant information available to local users in their language. Access is over the network through Brio.Portal.

## Data Sharing and Report Deployment

Bank2000 serves 30,000 named users, with an anticipated 1000 concurrent users initially, with peak use gradually going up to 15,000 concurrent users, as when users begin to log in regularly on Monday mornings to check postings made during the weekend or when users check postings for end-of-month, end of quarter, and end of year financials.

Users in each site will search, view, and download documents. A subset of these users will run jobs, such as financial reports. A smaller subset will be writing SQR and Brio.Query programs against which users will run their jobs.

Bank2000's Brio.Portal integrates with a data warehouse that stores BrioQuery and Excel reports. It also integrates with PeopleSoft Financials, providing a central repository and the ability to run PeopleSoft reports from each of the Bank2000 sites. Bank2000's Brio.Portal derives its user and category organization from the existing PeopleSoft hierarchical structures.

## Environment and Background

Bank2000 spans six European countries and the goal is to provide users access to documents in their own language. Each country, referred to as a *site*, has user accounts that include the locale (local language) as a property and the default directory of each user is locale-specific. For example, the default category of users in the Italian site is *Italiano*, which contains documents that are in Italian. Similarly, the default category of users in the English site is *English*, which contains documents that are in English. The intent is to make content accessible in the user's preferred language.

A technical team is responsible for planning, supporting, and growing the Brio.Portal services.

## Implementation Objectives

Bank2000 has installed Brio.Portal to meet the following objectives:

- *Organize users and categories by site* – User properties include a language locale, which identifies to the authentication server the site against which to verify the user account. The site category, which has an associated language local, is the default category of user accounts with the same language locale property.
- *Share documents* across sites. While a user has default category assigned according to site, individual users will have access to categories within other sites, according to permission levels.
- *Create and publish SQR and Excel programs* so that users can generate and run reports.
- *Run reports* as needed; some reports needed across all sites will be run as batch jobs.

## Implementation Considerations

Primary considerations for the Brio.Portal installation included the following:

- *Local access:* All users connect to the system from their desktop computers through a local area network.  
Site analysts who publish information to Brio.Portal regularly access other site categories. End users primarily search the category for local site, but occasionally browse other sites.
- *Access from remote sites:* Any users with IP addresses can have dial-up access.
- *Security:* Brio.Portal will limit access to ONE/Repository objects through user permissions.
- *Fault tolerance:* The Bank2000 Brio.Portal has two domains that have identical user accounts, groups, categories, and repository content. Contents in the two domains are synchronized at regular intervals. Brio.Portal IMPORT and EXPORT utilities provide for the initial loading of content and the continuing synchronization efforts.  
The user accesses the Brio.Portal system through ONE/WebClient. While each site connects to the service broker of a default domain, the configuration allows ONE/WebClient to connect to the other domain through another service broker connection, without the user being aware of the switch.

## Brio.Portal Usage Pattern

Users generally connect from their desks to search and download documents and run reports. Some reports are updated daily, others weekly, twice monthly, monthly, or quarterly.

Each site has designated report publishers.

The following table shows the Brio.Portal profile for Bank2000.

**Table 8** Brio.Portal Profile

<b>User accounts</b>	30,000
<b>Concurrent users anticipated</b>	1000 initially
<b>Geographical distribution</b>	6 sites, geographically distributed
<b>Users logging in per minute</b>	Unknown at this time
<b>Reports to be generated per hour</b>	10 per user
<b>Pages of output per report</b>	1-10; quarterly and annual reports will be larger (some over one hundred pages)
<b>Estimated number of objects in Brio.Portal</b>	1,000,000 initially

## Deployment

The Brio.Portal components deployed in the Bank 2000 Brio.Portal configuration are the following, as shown in Figure 5:

- Brio.Portal service agents: ONE/NameServer, ONE/AuthenticationServer, ONE/Repository, ONE/Event Server and ONE/ServiceBroker on a UNIX machine.
- One database server with Oracle 8.
- ONE/WebClient and ONE/ServiceBroker on Web servers at each site. Having a ONE/WebClient and a ONE/ServiceBroker at each site prevents bottlenecks from occurring during peak traffic, since tests demonstrate that bottlenecks first at ONE/WebClient and next at ONE/ServiceBroker. If the default domain is not available, then ONE/WebClient automatically connects to the other domain through a ONE/ServiceBroker in that domain.
- ONE/JobFactory replicated twice (three job factories) on a *reports* server that hosts Excel, SQR, and Brio.Query. One job factory is configured for running small jobs. The second job factory runs SQR and Brio.Query jobs in batch mode and the third job factory runs Excel jobs in batch mode.
- ONE/Administrator, desktop version, and a ONE/Service Broker on one client Windows NT machines.

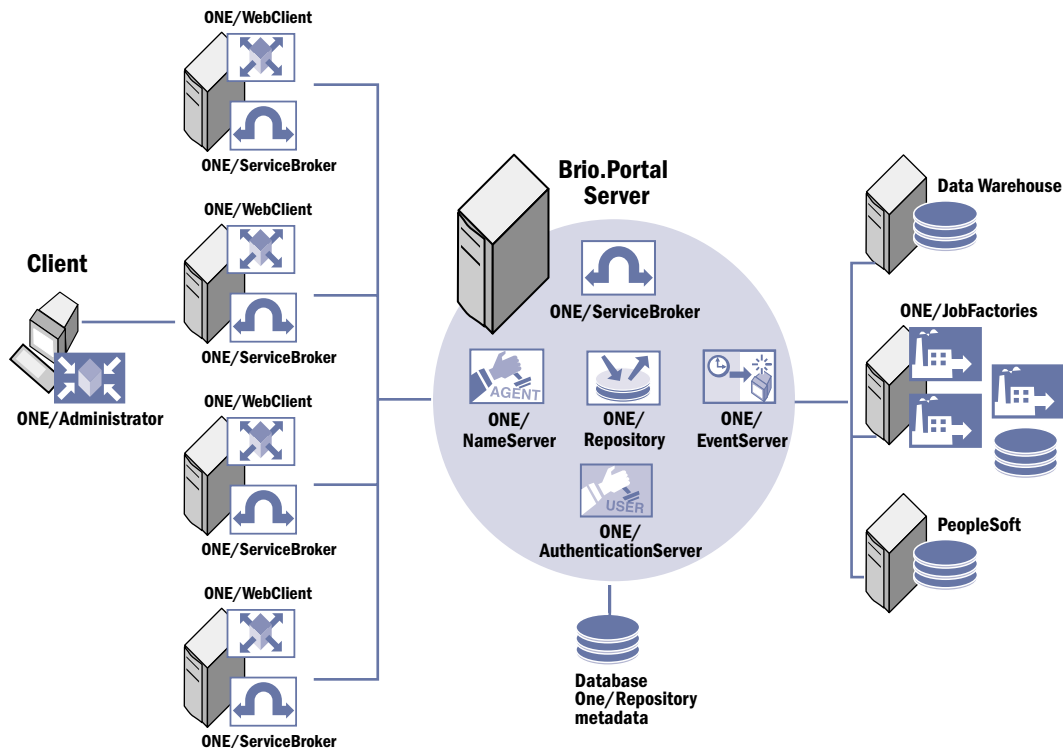


Figure 5 Bank2000 Brio.Portal System

The Bank2000Brio.Portal provides TCP/IP connection between the Web servers and the Brio.Portal servers. Logging in, repository search, as well as user execution of reports are done through the ONE/WebClient interface and ONE/ServiceBroker performs session management.

### **Pre-Installation Requirements**

Prior to Brio.Portal installation, the project database administrator set up a Brio.Portal account on the database server and created tablespaces for the persistent storage that Brio.Portal requires.

The Brio.Portal planners and administrator developed a Brio.Portal user profile based on existing user accounts from other systems and assigned a default group and default category for each user, by site.

### **Brio.Portal Persistent Storage**

Brio.Portal requires a database to store its persistent data. This persistent data consists of Brio.Portal service agent information, end user account information, and the reports and related objects published in Brio.Portal, together with metadata for these objects. The Bank2000 database administrator created the Brio.Portal database and a user account for Brio.Portal.

### **Access to Project Data**

Access to company data is provided through SQR, Excel, and BrioQuery. SQR and Excel are on a database server together with a ONE/JobFactory to run reports directly against the database.

### **Users and Groups**

Brio.Portal requires a password and a default group for each user. In addition to the default group, the user can be a member of other groups. User account names are those currently used at each site prior to the installation of Brio.Portal. These do not follow any specific convention. Any changes to userid or password will be done in the site directory, not from Brio.Portal.

Default groups are assigned according to site categories.

### **Brio.Portal Category Structure**

There are six main subcategories under the root. Below these, the number of subcategories depends on the structure of the content at the site. Currently, the following is the deepest structure:

- Bank2000
  - Headquarters
    - Checking
    - Savings
    - Loans

Selected links to other categories and specific reports are available within some categories. In addition, each application area has a link to the application Web site for documentation.

## Scaling Brio.Portal

Your company might start with a Brio.Portal implementation used in one department and then grow the system for use by the whole company. Or your company's initial implementation might be a large one, available to all geographically distributed branches of the company. In either case, you can follow the guidelines and recommendations in this section for distributing your Brio.Portal software components.

Before making any configuration adjustments to your Brio.Portal system to improve performance, you must first examine your system to determine where the performance bottleneck lies.

- If Brio.Portal is CPU-bound (consistently using more than 75% of the available CPU cycles), first try to increase the number of processors on the machine, if possible. If this is not possible, then try distributing and/or replicating service agents on multiple machines. ONE/WebClient, ONE/ServiceBroker, ONE/Repository, and busy ONE/JobFactories can each use a lot of CPU cycles and therefore, when processor utilization is high, isolating these service agents from one another will improve performance. You can replicate ONE/WebClient, ONE/JobFactories, ONE/ServiceBrokers, and ONE/AuthenticationServers on additional machines to distribute the load.
- If Brio.Portal is memory-bound, consider adding more memory before distributing agents.
- If the bottleneck is disk accesses, first make sure that data are striped across multiple disks, preferably 4 or more. RAID 0 (pure striping) should provide the best performance. RAID 0+1 (or RAID10, mirrored striping) will provide the best performance with redundancy, but is expensive to implement due to the number of disks required. RAID 5 provides redundancy, but results in poor performance for disk writes, common with running jobs and publishing results. Separating the Brio.Portal RDBMS and possibly the ONE/Job Factory from the ONE/Repository can help reduce performance problems due to disk bottlenecks. If this does not eliminate the disk bottleneck, check with the manufacturer to see if there are any tuning parameters for your disks, or consider purchasing faster drives.

Remember that making adjustments to eliminate one bottleneck can result in a new bottleneck elsewhere in the system. For example, distributing agents to reduce a CPU bottleneck can create a network bottleneck as more data will need to be transferred over the network.

## General Recommendations

Keeping the information in the previous section in mind, use the general recommendations that follow to help determine the best configuration for your installation.

- When bulk loading objects into Brio.Portal, turn subscriptions off to speed up the loading process. Turn subscriptions on again when the loading is complete.

- When large numbers of objects exist in the system and the service agents are running with SUN's JRE, the ONE/Service Broker, ONE/Repository and ONE/WebClient can run out of memory. Allocating more memory to these Brio.Portal services (by adjusting minimum and maximum heap settings) will alleviate the problem but may cause other processes running on the same system (RDBMS or other applications) to run out of memory. While we always recommend running Brio.Portal on a dedicated distributed system, this becomes particularly important as the number of objects increases.
- For generating many reports, you can ensure scalability to meet these increasing demands by installing ONE/Repository on a different machine than the repository database. Because the ONE/Repository stores the information and reports generated by enterprise users, it performs better when it does not need to share resources with the database.  
This scalability step also improves performance for typical user activities, including logging in, browsing, viewing, and running reports.

---

**Note** For best performance, the database/application server used for the Brio.Portal repository should not be used when running SQR jobs.

---

- *Improve response times by separating the ONE/WebClient onto a Web server.* Greater numbers of users who are browsing, viewing, and logging in to Brio.Portal can be serviced more efficiently by replicating ONE/WebClient on dedicated machines. ONE/WebClients can be directed to specific ONE/Service Brokers, thereby distributing the session management workload. By doing so, we increase the number of concurrent users that can be accommodated by the Brio.Portal system.
- You can also improve performance by *upgrading the machines where the ONE/NameServer, ONE/Repository, and ONE/EventServer* reside. The increased traffic from larger numbers of concurrent users will also affect these services, and they perform better with greater system resources. Likewise, you can upgrade the machine where the database server resides to better accommodate the increased information storage.
- *Further enhance performance by partitioning users and assigning them to specific ONE/WebClients, by specifying the ONE/WebClient URL in the user's login.*
- *Improve response times for frequent report generation by adding ONE/JobFactories either on the same machine or on separate machines.* Deploying additional ONE/JobFactories increases Brio.Portal's capacity for processing report requests.

As more users are running adhoc reports, you can speed up report processing by replicating ONE/JobFactory. Because the ONE/ServiceBroker can load balance between replicated ONE/JobFactories, it can service client requests more efficiently. And as report processing demands increase, you can further speed processing by adding RAM to machines where ONE/JobFactory resides.

You can configure multiple ONE/JobFactories to handle the same kind of jobs, or you can configure different ONE/JobFactories to handle different types of jobs. The ONE/Service Broker dynamically routes jobs to the appropriate ONE/Job Factory, thereby providing

location transparency by making it easier to add/remove job factories in the future. The type of job and available ONE/JobFactory determines where the job runs. You can also set a limit to the number of concurrent jobs for a job factory. When the job limit is reached, the job factory will refuse additional jobs until one of the executing jobs will be completed.

The ONE/ServiceBroker implements load balancing and fault tolerance algorithms to dispatch jobs to ONE/JobFactories. In addition, ONE/Job Factory is highly portable and can easily be distributed over the network. Highly portable ONE/JobFactories enable you to run jobs where the data resides-therby reducing network traffic. Because of the flexibility of this architecture, whether you replicate and how you deploy ONE/JobFactory can significantly improve the performance of enterprise database access.

---

**Note** If after you add multiple ONE/JobFactories, CPU, and memory, performance on the name server and repository machine is not sufficiently improved, de-install the ONE/JobFactory that is on the “original” machine, so as to free up more resources for the ONE/NameServer and ONE/Repository.

---

- *For running smaller reports frequently, as well as periodically running larger reports, deploy ONE/JobFactories to handle each case separately.* You can replicate and deploy ONE/JobFactories that handle the smaller jobs, while also deploying ONE/JobFactories that are tied to the data sources used by larger jobs.
- RAID 0 (pure striping) should offer the best performance but does not provide data redundancy. RAID 1+0 (or RAID 10, mirrored striping) will provide the best performance with redundancy, but is expensive to implement due to the number of disks required. RAID 5 (striping with parity) is a compromise that will provide less expensive redundancy but with reduced performance. If RAID 5 is used, data should be striped across at least 4 disks.

## Sample Distributed Configuration

Figure 6 shows a sample configuration that uses an architecture with distributed configuration.

- One machine is reserved for the database server
- One machine isolates the ONE/ServiceBroker
- One machine isolates the ONE/NameServer, ONE/Repository, and ONE/EventServer.
- One machine isolates the ONE/AuthenticationServer
- One machine is dedicated to the ONE/JobFactory.
- One machine isolates ONE/WebClient.

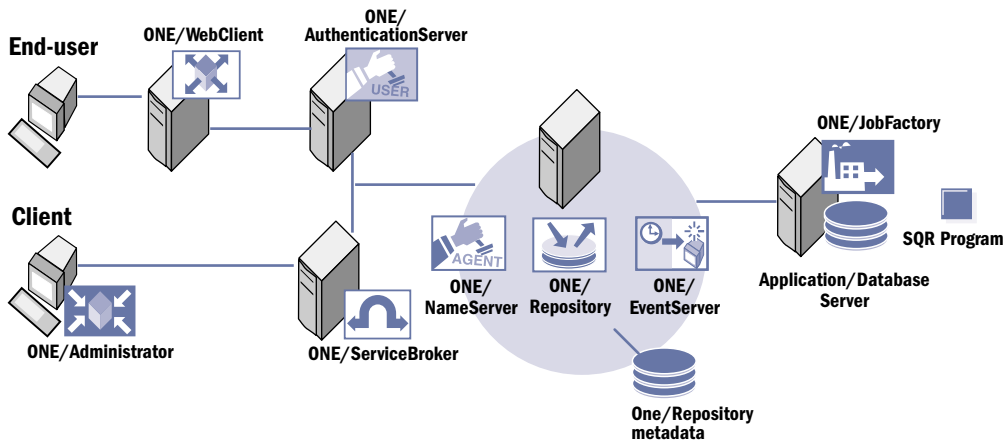


Figure 6 Isolating Service Agents to Meet User Needs

### Approaches to Replicating ONE/JobFactory

Intimately related to replication is how you deploy the replicated ONE/Job Factory. Two approaches exist: symmetric and asymmetric. You can use one or both depending on your needs.

**Symmetric Approach** Symmetric deployment means ONE/JobFactory is replicated and running on dedicated machines, as shown in Figure 5. Multiple ONE/JobFactories are configured to handle all types of jobs. When one ONE/JobFactory fails or is busy, new jobs are automatically routed to other ONE/JobFactories. The benefits of this approach are the load balancing and fault tolerance it offers.

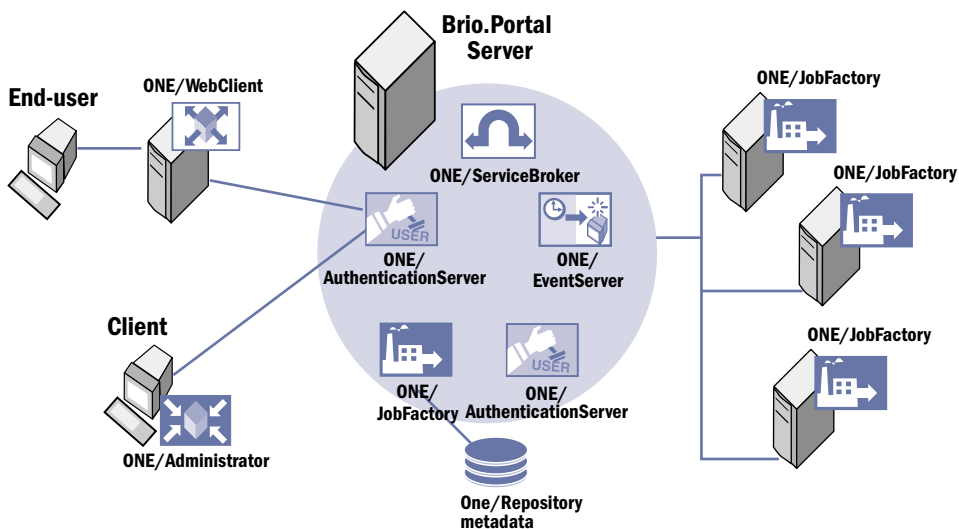
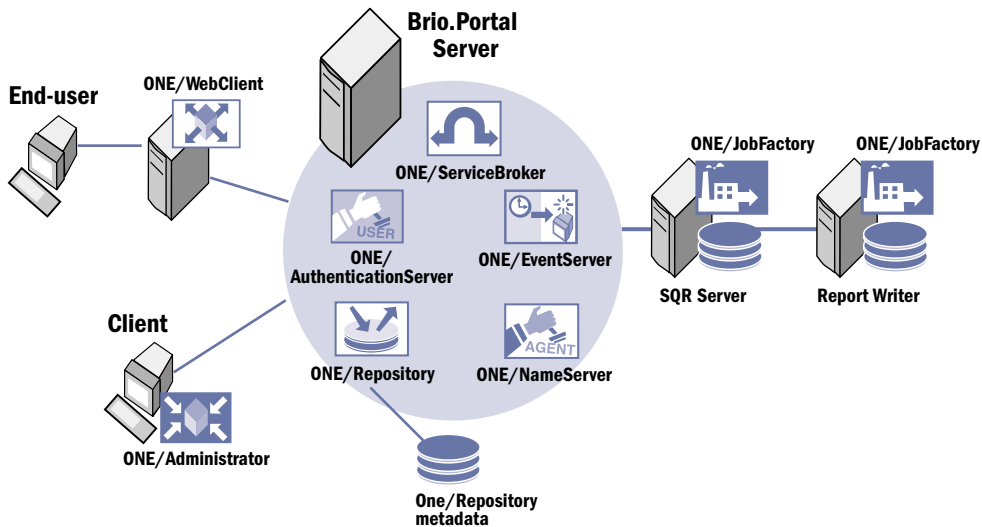


Figure 7 Symmetric Approach to Replicating Job Factories

**Asymmetric Approach** Asymmetric deployment means ONE/JobFactory is replicated and running on the same machines as the enterprise databases that they access. In this approach, shown in Figure 6, each ONE/JobFactory is dedicated to one data source and runs directly on the database server. The benefits of this approach are the improved performance and reduced network traffic that result from having the ONE/JobFactory on the same machine as the database server. If the nature of the data stored in the enterprise database includes large objects or accesses a great deal of data in each job, the asymmetric approach might be the best solution.



**Figure 8** Asymmetric Approach to Replicating Job Factories

**Using a combination of approaches:** Most enterprises experience a mixture of small and large reports. You can provide the optimal report processing performance by using a combination of symmetric and asymmetric ONE/JobFactories. For example, you can replicate and deploy ONE/JobFactory to process smaller reports that users run on an adhoc basis, as well as deploy ONE/JobFactory to process large batch jobs that typically run overnight.

## Distribute Users, Job Workload, and Brio.Portal Storage

As a final strategy for scaling Brio.Portal to meet enterprise demands, you can increase performance by partitioning Brio.Portal. You can:

- Assign users into groups that are serviced separately.
- Partition the job workload into separate ONE/JobFactories.
- Partition the Brio.Portal storage into multiple Brio.Portal domains.

### Distributing Users

You can distribute users to groups assigned to different ONE/WebClients. In turn, assign each ONE/WebClient to a ONE/Service Broker. Each group of users would use a different URL to log in.

This ensures that appropriate numbers of concurrent users are accessing each ONE/WebClient and ONE/Service Broker pair.

You can partition users by:

- Location (that is, either inside or outside of the firewall)
- Information category (for example, Sales or Accounting)
- Geography (for example, headquarters or Atlanta office)

## Partitioning Jobs

As mentioned previously, partitioning ONE/JobFactory can significantly increase performance. You can partition ONE/Job Factory in three ways:

- **By data source.** You can install a ONE/JobFactory where the data resides.
- **By job type.** Install each ONE/JobFactory with one or more report writing, query data analysis, or legacy applications. The ONE/JobFactory that is installed with the application will receive job requests that the application will process.
- **For load balancing.** Configure multiple ONE/JobFactories to process the same types of job requests.

## Partitioning the Repository

If your organization has many documents, reports, and programs to store and access, you can partition the ONE/Repository to improve access times-especially if your organization consists of multiple autonomous divisions that can easily store data separately. Because the Repository agent cannot be replicated, there are practical limits to how large the ONE/Repository can grow on a single machine. In contrast, a partitioned ONE/Repository allows for more database server machines to service information searches, retrievals, and jobs.

Each such partition is called a domain in the Brio.Portal architecture. Each Brio.Portal domain is a completely separate installation of Brio.Portal. At minimum, each domain includes one ONE/Repository, one ONE/NameServer, one ONE/AuthenticationServer, and one ONE/Service Broker. Each domain has its own users, information categories, and stored objects.

Consider implementing multiple Brio.Portal domains when a high correlation exists between information categories and user groups, or information categories and locations. For example, consider these Brio.Portal domains: Sales and Engineering. The Sales domain has several sales-related information categories and the Engineering domain has several engineering-related information categories. Searches and retrievals in each domain could be performed using separate ONE/Repositories.

Generally, each user will connect to only one domain when there are multiple Brio.Portal domains. However, if a user requires information from another domain, it is easy to switch over to that domain. When users login, ONE/WebClient allows them to choose the domain to which they want to connect. The logon screen presents the user with three fields: Username, Password, and Server. The Server field refers to the Brio.Portal domain and identifies the

machine where the domain's ONE/Service Broker is installed. Similarly, the ONE/Administrator allows the administrator to connect to any Brio.Portal domain on the network simply by pointing to the appropriate server.