Load Balancing

Service clients may create heavy load for service providers depending on the number of parallel running clients. A common way of distributing this load is using more than one node to host the services. Then, a load balancer will distribute the load among the identical services.The figure below depicts a scenario involving two servers and an external load balancer, for instance a hardware appliance. On each node, there is an independent Bridge installation. Each Bridge installation leads to two operating system services (Windows services, Unix daemons):

- **E2E Console** service: This service governs all deployed services (= online services) and also the E2E Proxy.
- **E2E Proxy** service: This is an Apache reverse proxy, configured and started/stopped via the Bridge.

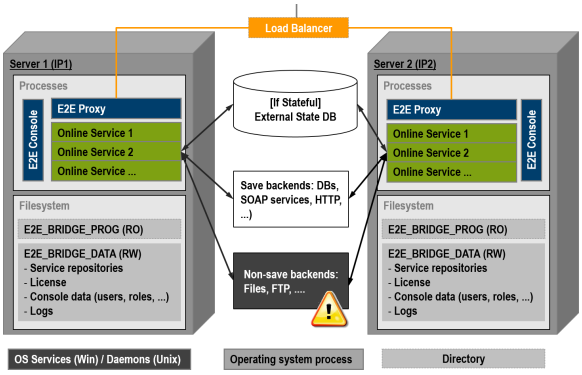
From an administration point of view, the E2E Console service is the only service that must be monitored and started by the operating system and possibly monitored by an external console such as HP Open View.

When a service is deployed on a server node, the Bridge starts an xUML Runtime instance executing this service as an operating system process and starts monitoring this service (see Monitoring Load Balanced Nodes). The deployed service (= repository) is stored in the **E2E_BRIDGE_DATA** directory. Since both server installations on both nodes are independent of each other, each service repository must be deployed on both nodes separately. This approach has the benefit that it is possible to update services on the fly by directing the load balancer to one node while updating the other node with a new service version. This way, the service is online all the time without any interruption. However, the drawback of this approach is that all deployments must be done twice.
If the services hold state either by using E2E Persistent State objects or by sharing persistent data, the data should be put into a shared external database. Of course, this makes this database a single point of failure. However, this is a common scenario most operation departments are used to.

As identical services are concurrently running, all write operations of the same services must use resources being safe of concurrent write operations. This is guaranteed by databases, message queues, and SOAP/HTTP services, etc. However, in general it applies not to file systems or (s)ftp. For such non-save resources, the modeler has to provide its own co-ordination mechanism, for example by a persistent state object controlling the access of the shared resource by the concurrent services. In the following figure, non-save resources are flagged with a warning sign. We recommend not using non-save backends at all in a load balanced set up.

Though this is not always feasible, in many use cases it makes sense to use the load balancing architecture for online services, but another configuration for batch processes that typically have to access files and file protocols (ftp, sftp, ftps, ...). This use case is discussed on Batch Processing.
A big advantage of a pure load balanced approach is that the Bridge processes do not hold any state. This implies that this approach scales very well and does not require any special load balancing features (even simple DNS round robin works quite well).

Figure: Two Servers and an External Load Balancer



On this Page:

Related Pages:

- Batch Processing
- BRIDGE Installation Guide
- Monitoring Node Instances
- Monitoring
- Persistent State Ownership

File System Setup

The file system contains two directories on each machine:

- **E2E_BRIDGE_PROG**: It contains all binaries for the Bridge. Read only access at runtime. It is updated only for patches and upgrades (see BRIDGE Installation Guide _).
- **E2E_BRIDGE_DATA**: This folder contains all deployed service repositories, service settings, and service log files; all Bridge data such as users, roles, groups, and server domain nodes; environment settings (e.g. **ORACLE_HOME**, ...); all Apache proxy configurations, and the proxy logs.

Maintenance

When maintaining one node, following steps must be taken:

1. The load balancer has to direct all new requests to one node.
2. After that, the node to be maintained can be shut down. This is done by stopping the OS services **E2E Proxy** and **E2E Console**. The shutdown of the Console process triggers the shut down of all deployed services. Each service being shut down will wait until it finishes the current request.

   > If one of the services cannot be properly shut down because of - for example - a hanging database connection, this process must be killed using the Bridge.

3. Do the maintenance work and all tests being required.
4. Re-deploy services that changed while maintaining the node – if necessary.
5. Start the Console service/daemon and then all required services (for those not having set the **Automatic Startup** flag).
6. Tell the load balancer to balance the load again.

> All online services should be managed by the Bridge. It is technically possible to start and stop all online services by using OS scripts. However, the Bridge is the only entity that knows if services are newly deployed or deleted. External tools do not. So, if starting up or shutting down the services (via **Automatic Startup** flag) is to happen automatically, the operator should start/stop the E2E Console services/daemon only.

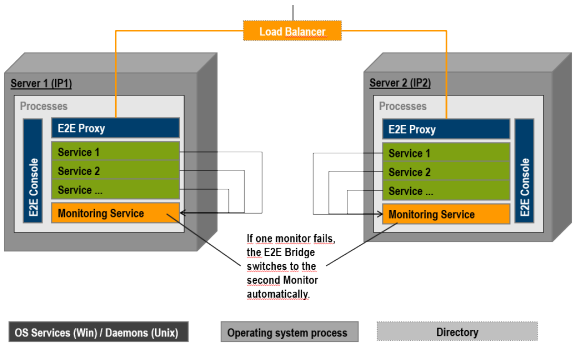
Additionally, it is important that operators monitor the Console service/daemon, because if it is dead, no management and monitoring of the online services will take place.

Monitoring Load Balanced Nodes

The Bridge monitors all deployed services. If a service writes a log entry of type ERROR or FATAL, or if a service terminates unexpectedly (crashes), the Bridge can call a monitoring service with all information found in the log file.
A monitoring service is a plain SOAP service implementing a given interface whose URL is registered at the Bridge (details see Monitoring Node Instances). Building monitoring services is on Monitoring pp.

Actually, the Bridge may register two URLs, the primary and the backup monitoring service. If the primary monitor fails, the backup will take over. In the load balancing scenario, the monitoring services on each node backup each other typically.

Figure: Monitoring Load Balanced Nodes

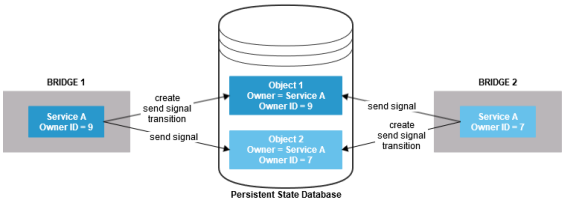


Load Balanced Persistent State

To setup a load balanced persistent state engine, you need to do the following:

- **Deploy the same persistent state service multiple times with a shared persistent state owner name.**
  You can either deploy the same service to divergent Bridges, or deploy the same service to the same Bridge under a different name. In the latter case, you need to assign the same owner name to both services, so they can share persistent state objects (see picture below). The owner name of a persistent state service can be changed in the persistent state settings of a service.

- **Setup the services to share the same external persistent state database.**
  You need to setup the persistent state components for an external storage medium (see Persistent State Components) and all services should use a matching database connection string.



Both services will create and process their own objects. These objects are identified not only by their primary persistent state key, but also by an owner name and owner id reflecting the actual service that owns these objects.
However, each service will be able to list and send signals to objects owned by the other service (having the same owner name).

> If one of the services (e.g. Service A with owner ID 9) is stopped, all objects with owner ID 9 will not be processed anymore (transitions, do activities).
> But:
>
> - All objects will still be visible for other services.
> - All objects will still be ready to receive signals. The signals will not be processed, however, but queued for later processing.

The redundant Service A with owner ID 7 can take over the processing of the persistent state objects, but will not do this automatically. To enable Service A with owner ID 7 to identify the persistent state objects to process, you need to change the owner ID of the objects from 9 to 7 in this case.
This can be done on the **Persistent State** tab of the Bridge with button **Manage Ownership**, see Persistent State Ownership.

Besides changing the owner ID of a redundant service to trigger processing of the stalled objects again, you can do the following:

- Restart the stopped service.
- Transfer the service to another Bridge (e.g. by service export and re-deployment).
  In this case, you also need to transfer file **PersistentState.tab** from the service directory (<your Bridge data directory>/<service directory>) to the new location. This file contains the actual owner ID of the service. If not present, a new owner ID will be assigned and all objects with the old owner ID will still be stalled.