SAP ARCHITECTURE BLUEPRINT

VERSION: 1.0

DATE: MAY 25, 2008

AUTHORS:

Julio C. Navas, Ph.D.

Live Enterprise

PROJECT NAME/CPROJECT TITLE: **LIVE ENTERPRISE**

SPONSOR/PROJECT INITIATOR: **RONALDO AMA**

PROGRAM/PROJECT LEAD: **JULIO C. NAVAS, PH.D.**

LEAD ARCHITECT: **JULIO C. NAVAS, PH.D.**

DEVELOPMENT: ☑ SAPLabs, mainly in Palo Alto

☑ Partner/ISV Exigen in Riga, Latvia

Document History		
Version	Date	Status (Comments)
1.0	May 25, 2008	Julio C. Navas, Ph.D.

1 Market and Product Background of Project/Program

Planned release date:	1Q 2009	
Underlying SAP NetWeaver release:	None	
Used SAP NetWeaver stacks:	☐ ABAP	☐ J2EE/Java EE 5
Use cases targeted by the project/program:		
Live Enterprise is SAP's vision for collecting, filtering, managing, aggregating, propagating, and publishing business events in a consistent, contextual way in an open and comprehensive framework, across multiple servers, locations, and companies.		
Strategic goals SAP wants to achieve with the project/program:		
Interviews with customers show that new problems have arisen as a result of SAP's success. It is common for customers to have 10's, 100's, or even 1000's of enterprise systems throughout their IT landscape. At the same time, the operational data generated by these systems, by partner company enterprise systems, and by complementary data systems, creates situations where it is not possible to insert this information into a central database in a timely manner. Worse, in a world where 73% of companies out-source significant parts of their business process, much of the operational data is not even directly available. Even if this operational data could be inserted into a central database, competitor best-of-breed systems are able to configure new event queries 3X faster than SAP from these multiple systems.		
Mandatory software capabilities to address goals, use cases, and target market:		
The software needs to be able to meet the following: • **Real-time Push of Information to Business Users** - Events across high volumes and distributions of data in "internet time." For example, up to 30,000 event correlations per second. • **Cross-enterprise Automation** - Automate the process of resolving new event requests to reduce the time-to-value for a new event question from the current industry standard of 5-6 person-weeks per question. • **System Scaling** - Throughput would be independent of data volume, quantity, and distribution pattern. For example, 2000 tier-one suppliers and 30,000 tier-two suppliers. • **Manage the data "in place"** - But easily allow the flexible and ad hoc movement of data when necessary • **Allow local control over data administration** - But easily enable participation in broader data footprint of "integrated enterprise" • **Layer on top of Existing Landscape** - Within and across enterprises through standards already in place		

2 Architecture

2.1 Main Architecture Decisions

Live Enterprise (LE) is designed to be a horizontal event enabling solution for all industries and companies that have a need for a global and unified view into their operational enterprise data no matter how widely distributed.

Since LE is meant to answer event-based operational questions in near-real-time, it needs to correlate raw operational data on-the-fly. Customer interviews indicate that within 2-3 years, it will be necessary to perform up to 30,000 event correlations per second. Therefore, LE chose to use third-party complex event correlation engines that can handle huge volumes of data streams in real-time. LE currently uses the Coral8 Engine that can handle 50,000 event correlations/sec and 300,000 event filterings/sec.

Currently, a major impediment to eventing systems is the large amount of time required to program the logic behind each operational question. In other words, not do users need to specify “what” they want, they also need on average 5-6 weeks of programming to specify “how” to go about answering the question. LE is therefore designed with federation and streaming database principles in mind so that a user need only specify “what” they want and LE automatically determines “how” to go about answering the question. One of these principles is the abstraction of data organization and formats through the use of a single normalized and global view into all of the operational data.

The raw information that LE needs to correlate is often found scattered over multiple enterprise systems and across multiple locations. Customer interviews have shown that the number of systems could be as high today as 3000+ enterprise systems. Since 73% of companies out-source major portions of their business process these days, oftentimes, the operational information is also scattered across multiple companies. These partner companies form a value chain with multiple tiers. Customer interviews indicate that these tiers can be as large as 2000 companies in tier 1 and 25,000 companies in tier 2. As a result, in many customer instances, this raw operational data is too voluminous to be centralized. Therefore, LE is designed to be able to operate in a distributed manner and as close as possible to the systems of record themselves. Also, LE is designed to scale to very large numbers of servers, locations, and companies and allow them to operate in an uncoordinated and decentralized manner. It accomplishes this through the use of a network routing layer that is designed to operate in the same manner as the Internet.

LE needs to effectively layer on top of an existing enterprise landscape. However, customer interviews show that all of these potentially thousands of enterprise systems can be comprised of SAP systems, competitor offerings, and custom-built applications. Therefore, LE is designed to operate across a heterogeneous landscape.

Finally, LE is meant to be used in an application environment. Therefore, LE uses a publish-and-subscribe paradigm with standard application APIs.

2.2 Main Architecture Concepts (run-time)

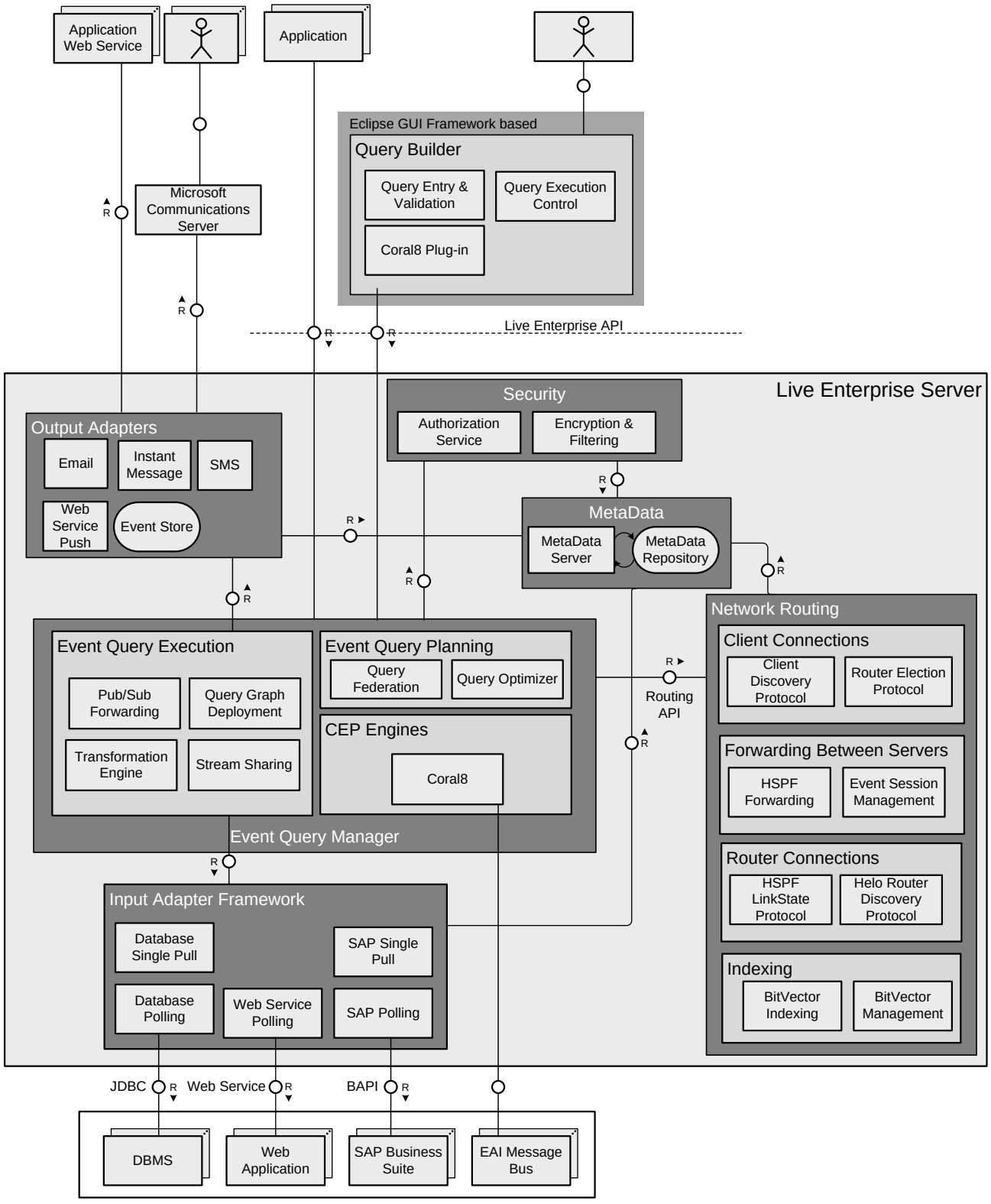

Figure 1 - Live Enterprise Run-time Architecture

2.2.1 Event Query Manager

The Event Query Manager is the central hub for the LE system. It will plan event queries and coordinate the execution of the event queries.

2.2.1.1 Event Query Planning

When a client submits an event query for processing, an execution plan will first be created. The event query will be analyzed, optimized, and federated into simpler fragments composed of individual raw event types and tables. It will use the global⇔local mapping information in the MetaData Server to determine what transformations need to be performed. It will use the index information within the Network Router to determine whether it needs to execute the event query (or fragments) remotely.

2.2.1.2 Event Query Execution

Once the event query plan is finalized, LE will setup the paths down which the event query and results sets will flow. The query plan will be deployed amongst the underlying Input Adapters, Complex Event Processing engines, and remote LE servers. If LE detects that multiple users who have equivalent access rights are asking the same question, then stream sharing will be used and their query execution will be joined together.

All event queries are executed in a publish-and-subscribe manner and the Event Query Manager will act as a hub for information flowing between the adapters, CEP engines, and remote LE servers. If the information comes from an Input Adapter, the data will be transformed from that data source's local schema to the normalized global schema. The event query will be executed continuously until the user specifically unsubscribes.

2.2.1.3 Event Query Processing

The sequence diagram for the execution of an event query at a single LE server connected to one or more data sources is shown in Figure 2.

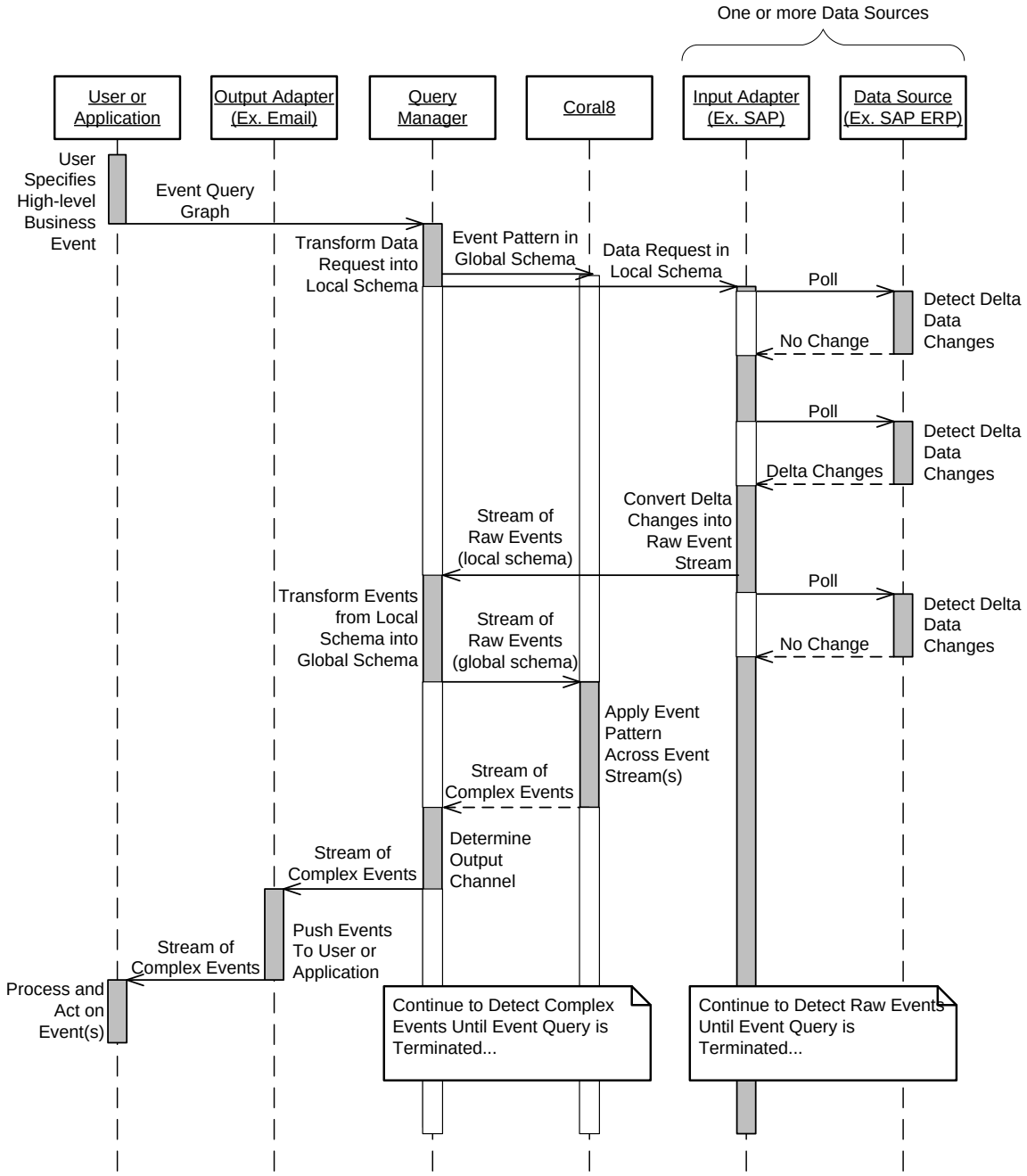

Figure 2 - Single Server Event Execution

2.2.1.4 Federated Event Query Processing

An Event Federation Grid will be used to achieve fast performance. Here we assume that the data sources will be distributed in a disjoint manner between several LE servers. These servers may or may not be in different locations.

Once the user defines the specific complex event pattern for the business event he wishes to receive, LE will analyze that event and attempt to break it up into smaller and simpler pattern fragments. If more than one data source applies to an event pattern fragment, then copies of that fragment would be sent to all LE servers that are connected to the applicable data sources and executed in parallel. Note that this process may be recursive. Event pattern fragments themselves may be broken up into even simpler patterns and also executed in parallel.

When the fragment's pattern is matched by the local data, then the result would be sent to the first LE server. This server would collate the responses from the various fragments and perform any last-minute calculations, joins, or functions. If, after all this, the server detects that the original event pattern has been matched, then the response would be forwarded to the user. See Figure 3 for a sequence diagram depicting this algorithm.

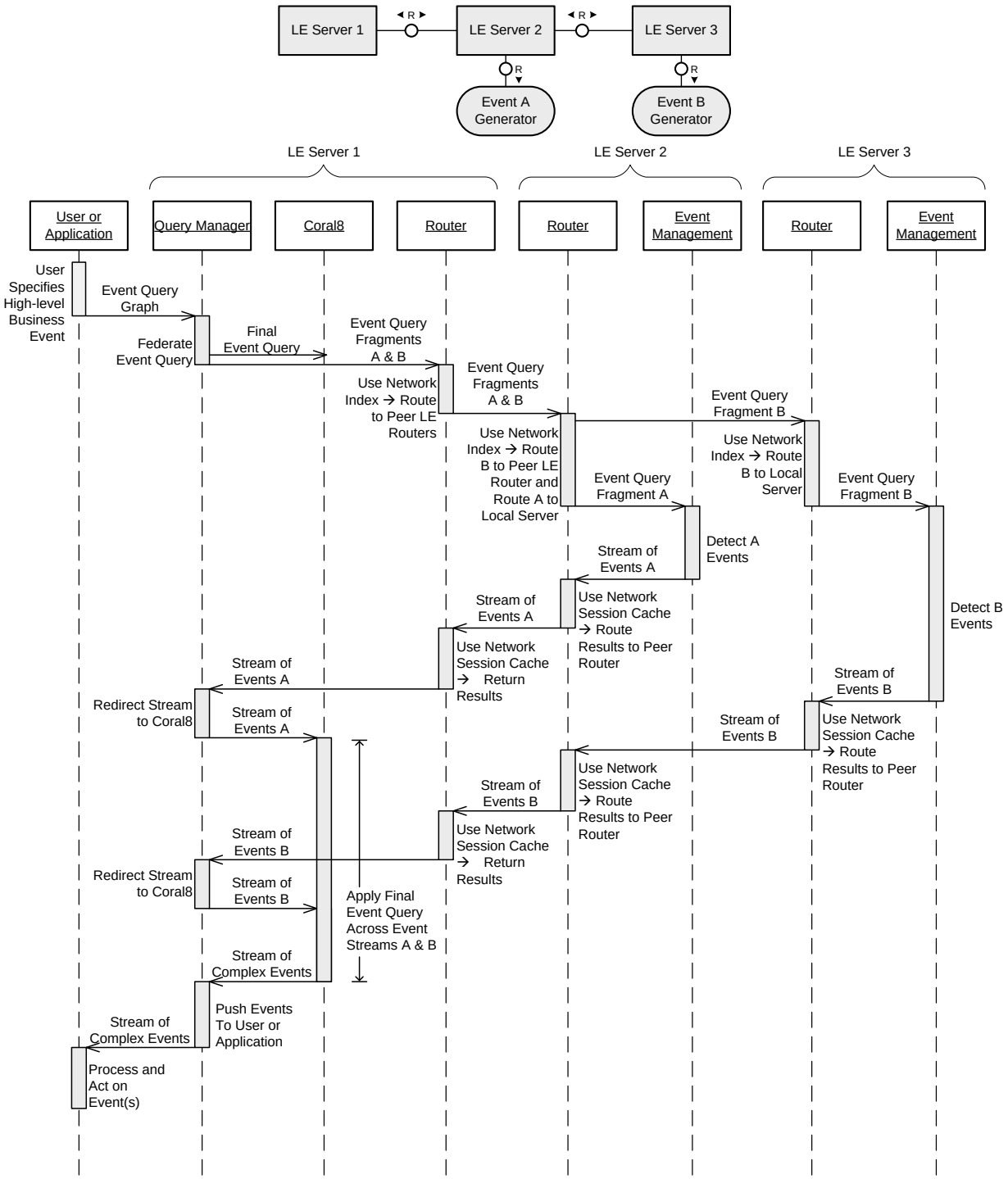

Figure 3 - Event Query Federation

2.2.2 Network Routing

Live Enterprise Network Routing uses a new link-state routing protocol which allows event queries and their corresponding responses to be transported multi-hop through a network from a sender to a set of destinations using a description of the receivers in the form of multiple arbitrary identifying descriptive names. Destinations can have multiple names and that these names can be acquired or removed in a dynamic fashion.

2.2.2.1 Indexing

In LE routing, the system is designed to allow computer hosts to have multiple descriptive names, such as the event types that they offer. Furthermore, these names or characteristics can be acquired or removed by a host in a dynamic fashion with little overhead. Using hash-based probabilistic equations, these names are encoded into bit vectors in such a manner that the bit vectors encode only the existence of the names. Because only existence information is stored, compression ratios of 200:1 have been achieved. The bit vectors can then be managed using standard vector mathematics so that they can be easily merged, compared, and their vector lengths normalized.

2.2.2.2 Hierarchical Shortest-Path First (HSPF)

LE uses a variation of the standard Internet link-state routing Open Shortest-Path First (OSPF) protocol. Use of an OSPF-derived protocol allows LE to have the following advantages:

- Automatically discover nearby LE servers and create connections between them using the OSPF HELO protocol
- Dynamically distribute network knowledge throughout the LE topology
- Dynamically adjust to network topology changes, such as server connections going down.
- Use of router election protocols so that each location can have a primary LE server and a “hot stand-by” in case of a failure of the primary server
- Use of hierarchical routing to scale to large numbers of destinations and to hide the organization of different hierarchy areas from each other (such as when different areas correspond to different companies)

However, the OSPF protocol has limitations. For instance, OSPF is a logical protocol that only understands network links and the computer nodes on those links and has no knowledge of the enterprise event and data content on them. LE extends OSPF by inserting content information in the form of bit vectors into the routing tables so that the LE routers understand the enterprise data that is available on those networks and computer nodes. Also, OSPF only allows two levels of hierarchy, which limits it to a theoretical maximum of 200 companies and 4000 destination computer nodes. LE extends OSPF further so that it can have an arbitrary number of hierarchical levels to reach many thousands of companies and destinations. The OSPF HELO protocol only creates automatic connections between routers. A variation of the HELO protocol is used by LE to automatically detect and connect routing clients.

2.2.2.3 Forwarding Between Servers

Using HSPF, each LE router will have a routing table entry for each destination in the network. Each entry will then contain information on a destination's event and data content, IP address, and the shortest path to the destination.

The destination for a message is defined as the event or data content that is sought. When forwarding a message, an LE router uses its routing table information to determine the specific final destinations that contain that event or data content and then forwards the message on the shortest path to those destinations. Because the destinations for a message are described using event or data content, this allows for a decoupling of event consumers and event producers, which, in turn, allows for new destinations, locations, or companies, to be added quickly in an uncoordinated fashion and without requiring that the system by reprogrammed in any way.

Once a message has been forwarded all of the way from the sender to all of the destinations, the routers will have created a one-to-many shortest-path routing tree with the “root” at the sender and the “leaves” at all of the destinations. This routing tree is called a session and soft-state at each router, in the form of a routing cached, is used to maintain and manage a record of this session.

Because probabilistic equations are used for the indexing for engineering trade-off reasons, false positives may result during the forwarding phase. Final checks at the destinations are performed and, if false positives are detected, those routing paths may be pruned as a result.

2.2.3 Live Enterprise API

LE has two standard interfaces for clients: a Java API and a Web Service API. The Web Service API, as currently implemented, is actually a wrapper for the Java API.

2.2.4 Query Builder UI

The query builder uses the industry-standard Eclipse GUI Framework based tool for construction, execution, and status tracking of queries. The query builder is used to create and validate event queries as well as to monitor event query execution.

2.2.5 Output Adapters

The Output Adapters are designed to give multiple options for forwarding results to the user. Result sets can be sent directly to an application via a web service, they can be stored for later perusal in an event store, and they can be sent directly to a user via a Microsoft Communications Server using email, instant messaging, or SMS to a cell phone. LE can also monitor a user’s online presence determine the right output adapter to use accordingly.

2.2.6 Input Adapter Framework

The Input Adapter Framework is designed to abstract away underlying data sources. Each adapter presents the same manager API to the Event Query Manager and registers with the Event Query Manager so that it knows about them. Adapters can be configured as either event sources or historical data sources. Currently, adapters exist for SAP systems, databases, and generic web services. The framework is designed as a Software Development Kit so that new adapters can be created in a straightforward manner. Adapters have a filtering capability that can be configured on a per-event-query basis.

2.2.7 MetaData Server

As the name implies, the MetaData Server and Repository contain run-time and configuration information about the LE system. For instance, as an example of configuration information, it will contain the global schema, the various local schemas, as well as their various mappings. In terms of run-time information, it can contain user defined event queries and LE network topology status information.

2.2.8 Security

LE Security uses a two-level security model: the global Live Enterprise level and the local Data Source Level. At the global level, LE can authenticate a user and enforce that user’s access rights and restrictions down to the row level. Also, event queries and their result sets are encrypted with a different key being used on a per query basis.

One of the goals of LE is to give data owners full control over the access to their data by others – this is especially important when dealing with information exchanges between companies. Therefore, LE leverages the underlying user access rights by identifying the user who is asking the event query to the underlying enterprise system. Based on that identification, the underlying system can then determine how best to answer the event question and what access rights to afford that particular user.

2.3 Main Architecture Concepts (design-time)

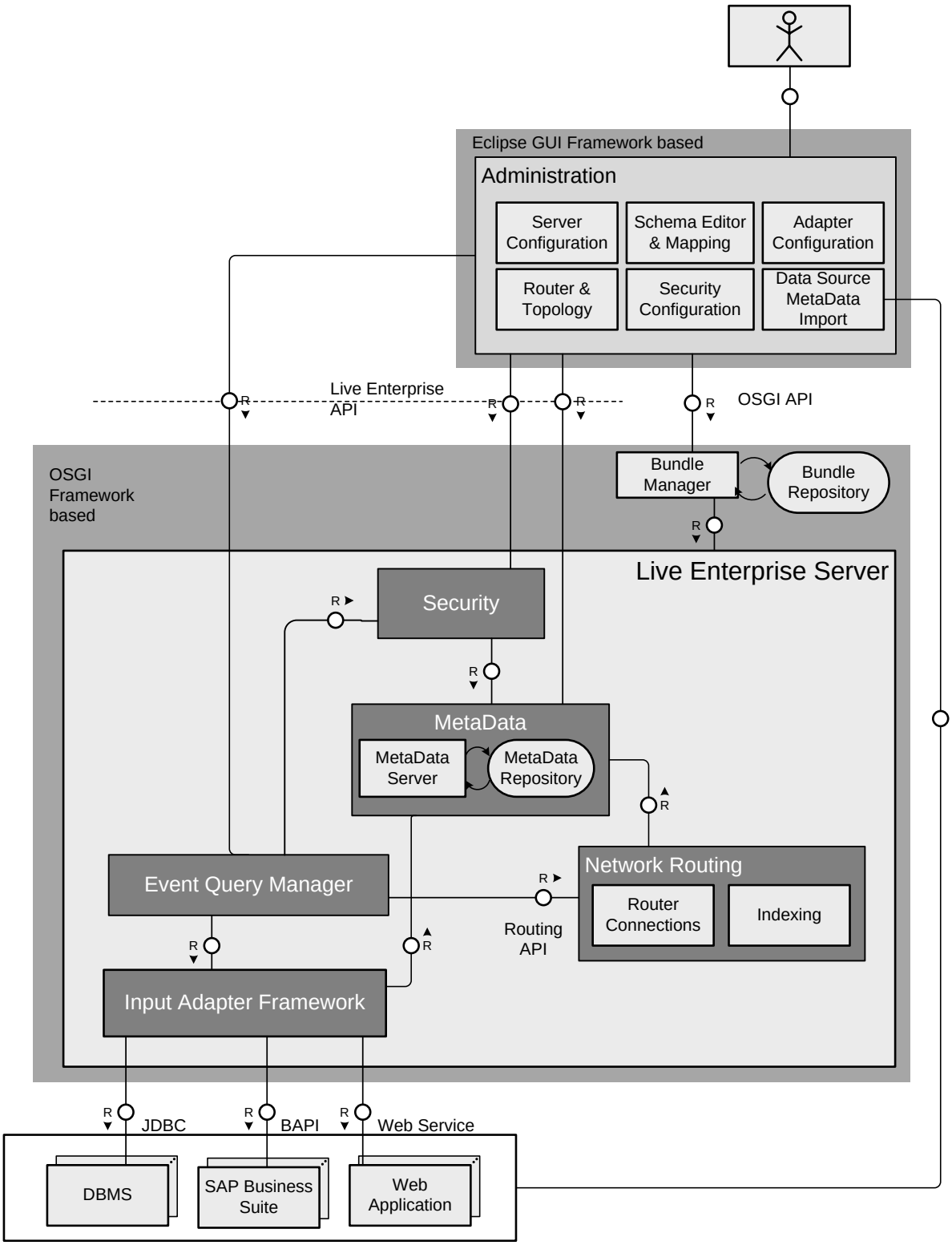

Figure 4 – Live Enterprise Design-time Architecture

2.3.1 OSGI Framework

Live Enterprise is an OSGI Framework based (OSGI Service Platform Specification R4 (http://www.osgi.org/)). This framework allows for all LE servers in multiple locations within an enterprise to be managed from a single

Unified Administration GUI. All Java server and UI side functionality is packaged into bundles and is managed by a Bundle Manager at each physical server. Furthermore, all bundles can be pre-loaded into a Bundle Repository and then distributed to all of the LE servers throughout an enterprise. See Figure 4.

2.3.2 Unified Admin GUI

The Unified Admin GUI is an Eclipse GUI Framework based unified administration tool, to which different functionality can be added via plug-ins. See Figure 4. Currently, there are plug-ins for a variety of functions such as:

- Managing LE bundles via OSGI
- Configuring routing connections and viewing their network topology
- Managing and creating input adapters and their connections to the underlying data sources
- Managing global and local schemas. This includes configuring event types and global ↔ local mappings as well as importing the metadata from data sources and converting them into local schemas.
- Managing security settings

2.3.3 Global Schema and Index Configuration

A sequence diagram showing how the Unified Administration GUI interacts with the LE server components in order to integrate a new data source into LE and populate its global schema and indexes is shown in Figure 5. Note that the sequence diagram shows the interaction between three instances of the LE server connected in series.

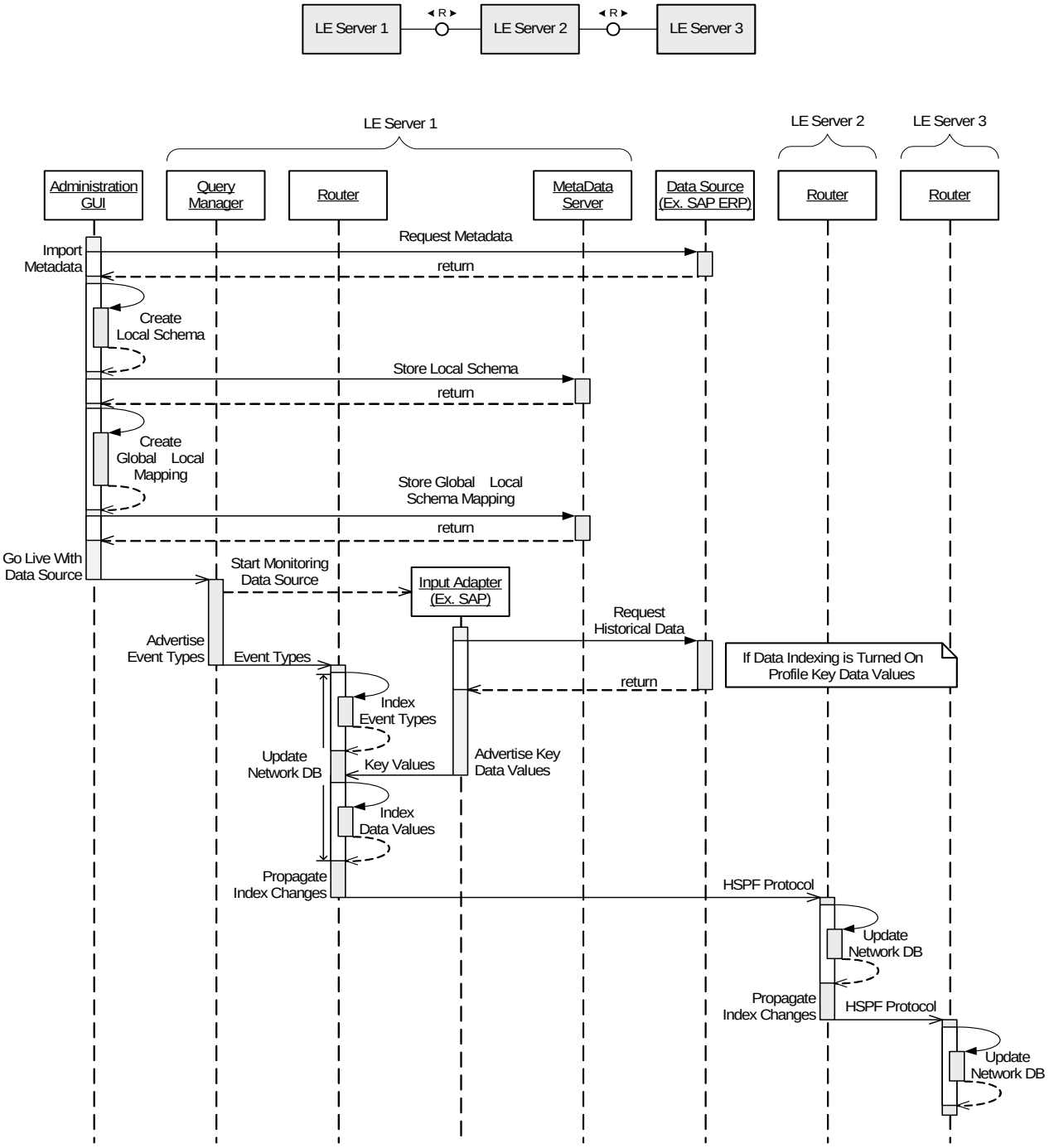

Figure 5 - Indexing and Global Schema

2.4 Total Cost of Ownership

Since LE management is based on OSGI, this allows for low TCO because only the bundles that are required for a particular location need to be installed. When an updated bundle is available, it can simply be loaded into the Bundle Repository and, from there, pushed to all LE server installations that require it.

2.5 Deployment

LE is designed to be deployed in a single location, multiple locations, and in multiple companies.

2.5.1 Single Location Deployment

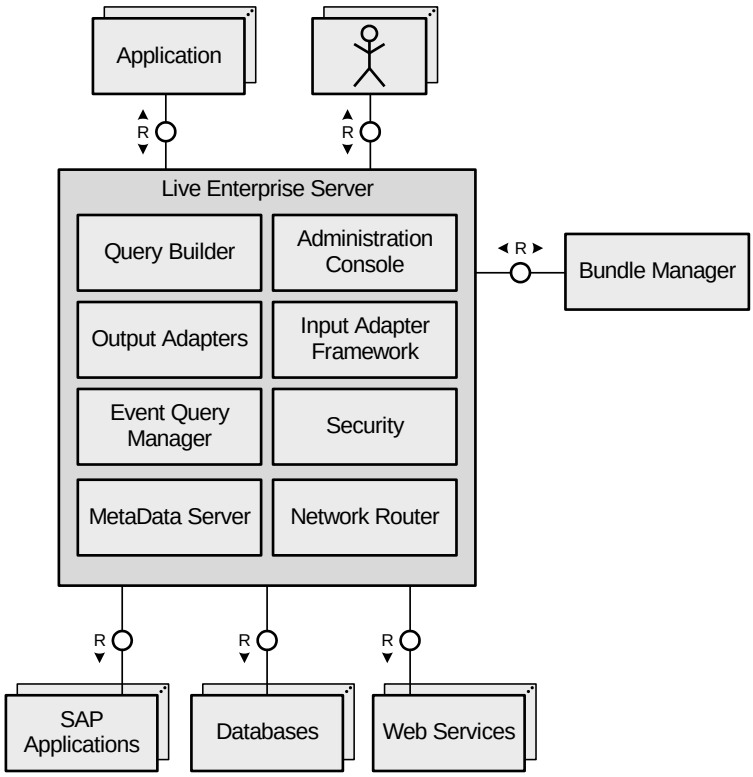

Figure 6 - Live Enterprise Deployment in a Single Location

LE can be deployed on a single physical server or on multiple physical servers. A typical installation will be on a single server. Deployment of the various LE components and the Bundle Manager on a single server is shown in Figure 6. Note that LE is mostly written in Java and requires a Java run-time engine to be pre-installed on the server. The Network Router is the only component written in C++ for performance purposes. The database required by the MetaData Server is installed automatically as part of the standard LE deployment.

2.5.2 Multiple Location Deployment

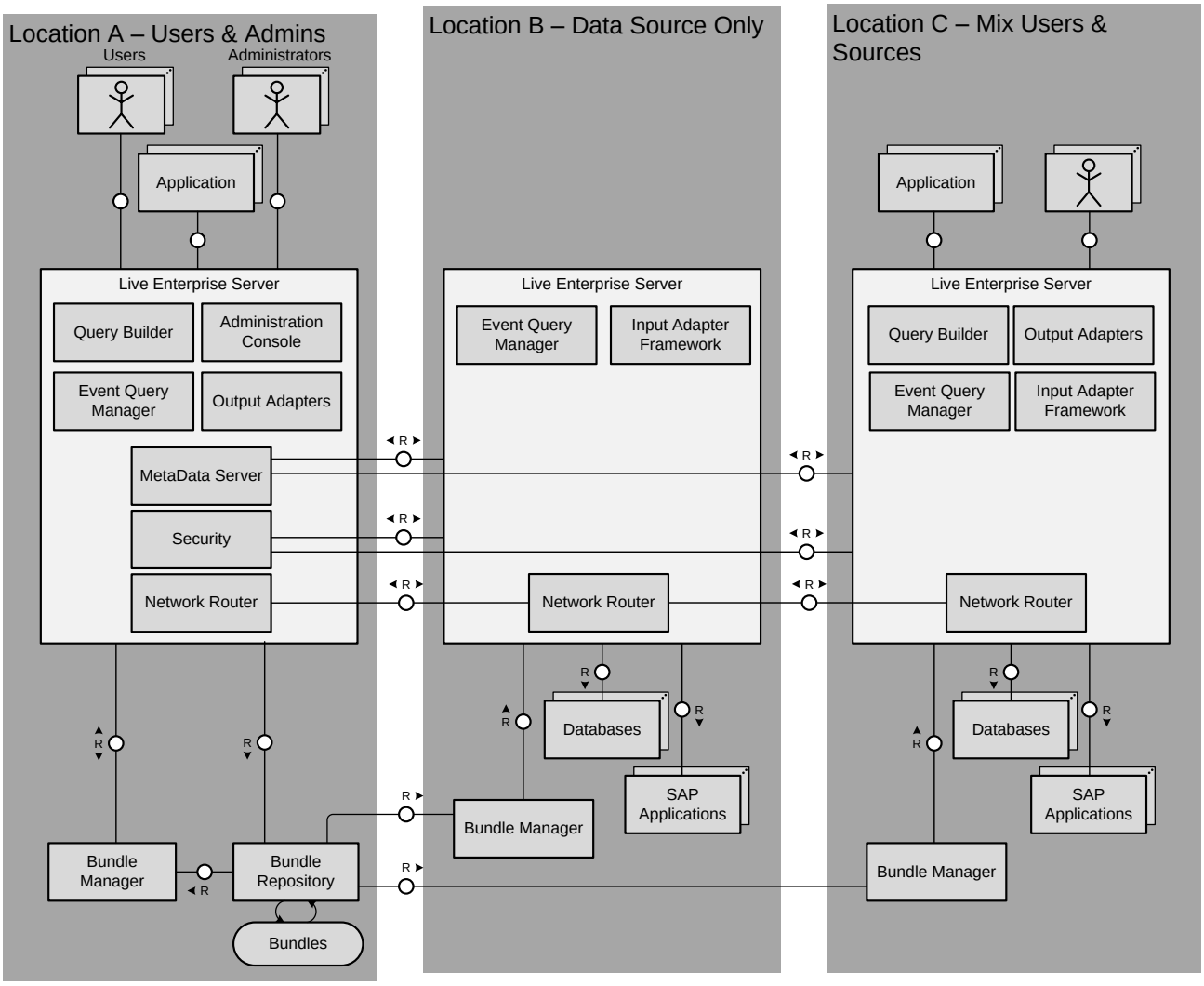

Figure 7 - Live Enterprise Deployment in a Multiple Locations

Each location does not need a full copy of all of the components of LE. An administrator would push only the necessary components from the Bundle Repository to the different locations. Note that a copy of the Bundle Manager must be installed at each location in order to manage the local components. See Figure 7.

In a multiple location scenario, the individual components are designed to automatically discover each other and establish connections between each other. However, the servers need to be on the same LAN subnet in order for the individual components to find each other. Otherwise, they must be specifically configured to connect to each other.

2.5.3 Multiple Company Deployment

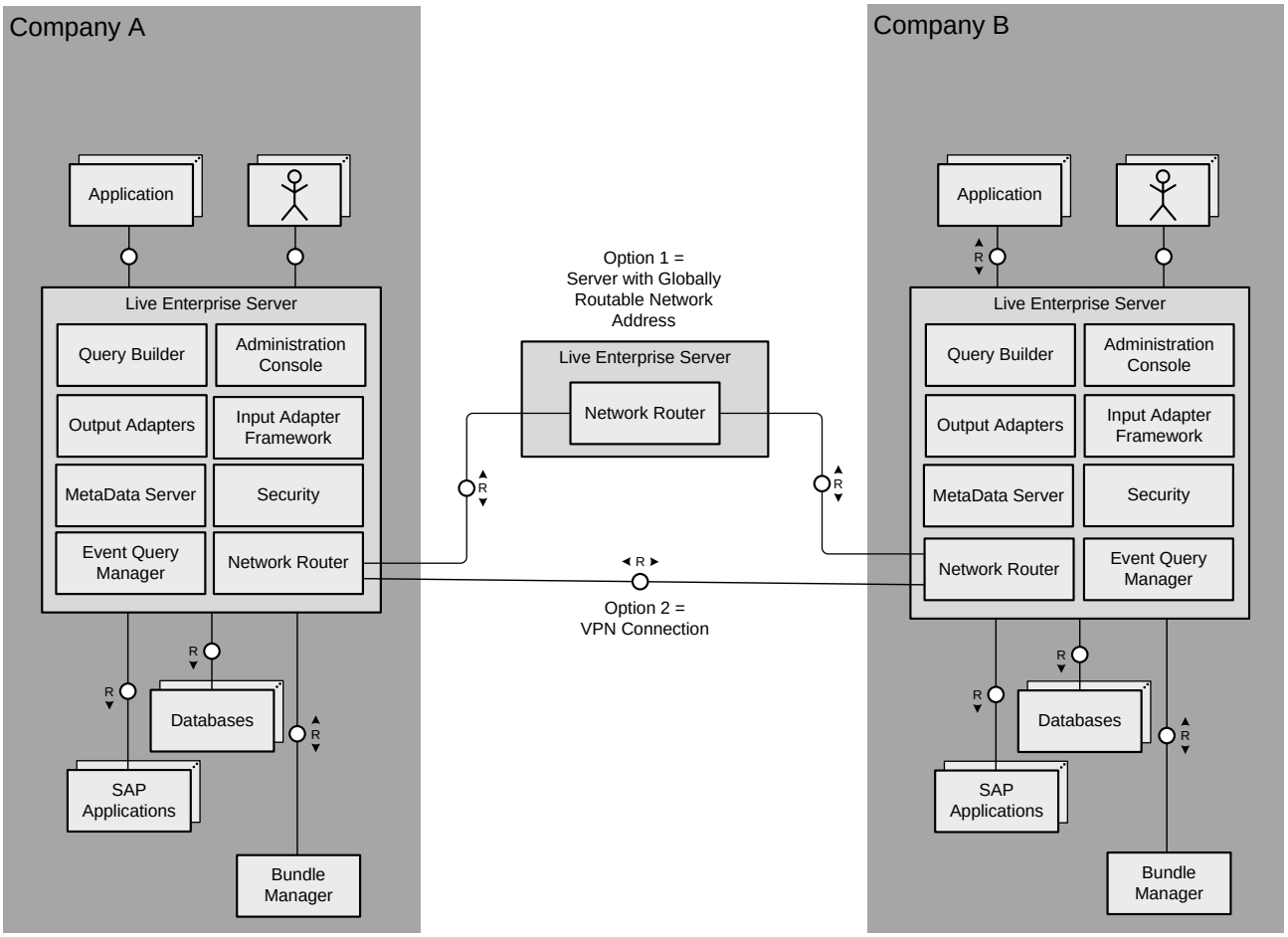

Figure 8 - Live Enterprise Deployment in a Multiple Companies

The key difference between a multi-location scenario and a multi-company scenario is the presence of firewalls between the companies. There are two options in order to bridge this obstacle: a secure point-to-point connection and a trusted third party. See Figure 8.

The two companies can create a secure point-to-point connection, such as a VPN tunnel, between the two companies. The LE installations within each company can then be joined together by connecting their Network Routers.

Since LE Network Routers are able to route multi-hop between them, it is possible to route event queries and result sets through a trusted third party. In this case, the third party at least has the Network Router component installed. Both companies would then make secure connections to the third party and connect their LE Network Routers to the third party's Network Router.

3 Open Issues, Outlook and Risks

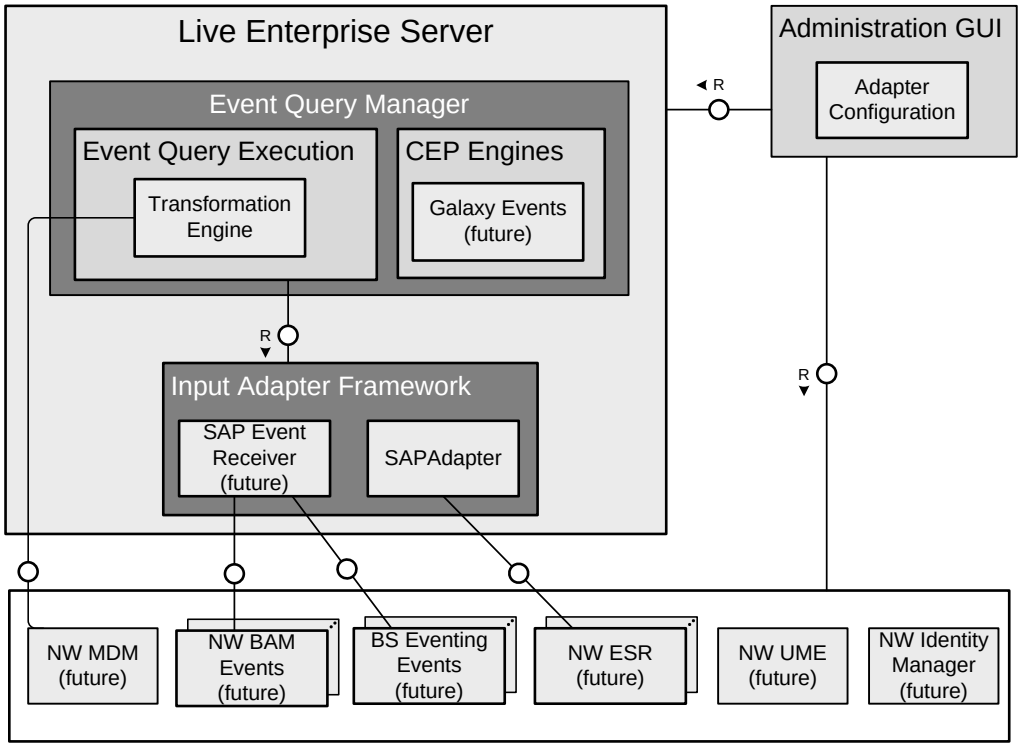

Figure 9 - Future SAP Connections

The LE server will in the future be further connected to the SAP NetWeaver infrastructure. Of particular importance will be the improved integration with current and future SAP event infrastructures. Both NetWeaver BAM, which is part of SAP NetWeaver 7.1, and the relatively new BS Eventing project would allow for events to be exported from SAP application systems and directly transported to LE without the need for polling. The Galaxy Project's event detection engine can be integrated into LE as a core event engine. In order to best determine the information that is available for event and data correlation, the Enterprise Service Repository can be used to make Input Adapter instantiation and configuration easier in an enterprise environment. In order to reuse and synchronize on user security settings, LE will be connected with the NetWeaver User Management system and with NetWeaver Identity Server. Finally, for those cases where multiple departments within an enterprise or multiple partner companies have different names for the same product, a connection between the LE Transformation Engine and the NetWeaver MDM product would allow for transformations based on synonyms and for the normalization of product names as part of the LE Global Schema.