

Linking Enterprise Business Systems to the Factory Floor

Kenn S. Jennyc

Information is the fuel that drives today's business enterprises. The ability to link different components in the enterprise together in a user-friendly and transparent manner increases the effectiveness of companies involved in manufacturing and production.

Computers have had a profound effect on how companies conduct business. They are used to run enterprise business software and to automate factory-floor production. While this has been a great benefit, the level of coordination between computers running unrelated application software is usually limited. This is because such data transfers are difficult to implement, often requiring manual intervention or customized software. Until recently, off-the-shelf data transfer solutions were not available.

HP Enterprise Link is a middleware software product that increases the effectiveness of companies involved in manufacturing and production. It allows business management software running at the enterprise level, such as SAP's R/3 product, to exchange information (via electronic transfer) with software applications running on the factory floor. It also allows software applications running on the factory floor to exchange information with each other.

HP Enterprise Link is available for HP 9000 computers running the HP-UX* operating system and PC platforms running Microsoft's Windows® NT operating system.

This article will discuss the evolution of the link between business software systems and factory automation systems, and the functionality provided in HP Enterprise Link to enable these two environments to communicate.

Background

Initially, only large corporations could afford computers. They ran batch-oriented enterprise business software to do payroll, scheduling, and inventory.



Kenn S. Jennyc

Kenn Jennyc is a software engineer at the HP Lake Stevens Division. He

worked on the software design, development, and quality assurance for the HP Enterprise Link. Before that he worked on software design and development for the RTAP (real-time application platform) product. He received a BSEE degree from the University of Calgary in 1983 and came to HP in 1989. Kenn was born in Calgary, Alberta, Canada, is married, and has two children. In his spare time he likes to fly his home-built aircraft and dabble in analog electronics.

As the cost of computing dropped, smaller companies began using computers to run business software, and companies involved in manufacturing began using them to automate factory-floor production.

Although factory-floor automation led to improved efficiency and productivity, it was usually conducted on a piecemeal basis. Different portions of an assembly line were often automated at different times and often with different computer equipment, depending on the capabilities of computer equipment available at the time of purchase. As a result, today's factory-floor computers are usually isolated hosts, dedicated to automating selected steps in production. While various factory-floor functions are automated, they do not necessarily communicate with one another. They are isolated in "islands of automation." To make matters worse, the development of programmable logic controllers (PLCs) and other dedicated "smart" factory-floor devices has increased the number of isolated computers, making the goal of integrated factory-floor computation that much harder to achieve.

While production software was generally used for smaller, more isolated problems, business software was used to solve larger company-wide problems. Furthermore, while

production software was more real-time oriented, business software was more transaction and batch oriented. These differing needs caused business systems to evolve with little concern for the kind of computing found on the factory floor. Similarly, production systems evolved with little concern for the kind of computing found at the enterprise level. As a result, many enterprise-level business systems and factory-floor computers are not able to inter-communicate. **Figure 1** shows an example of the components that make up a typical enterprise and factory-floor environment.

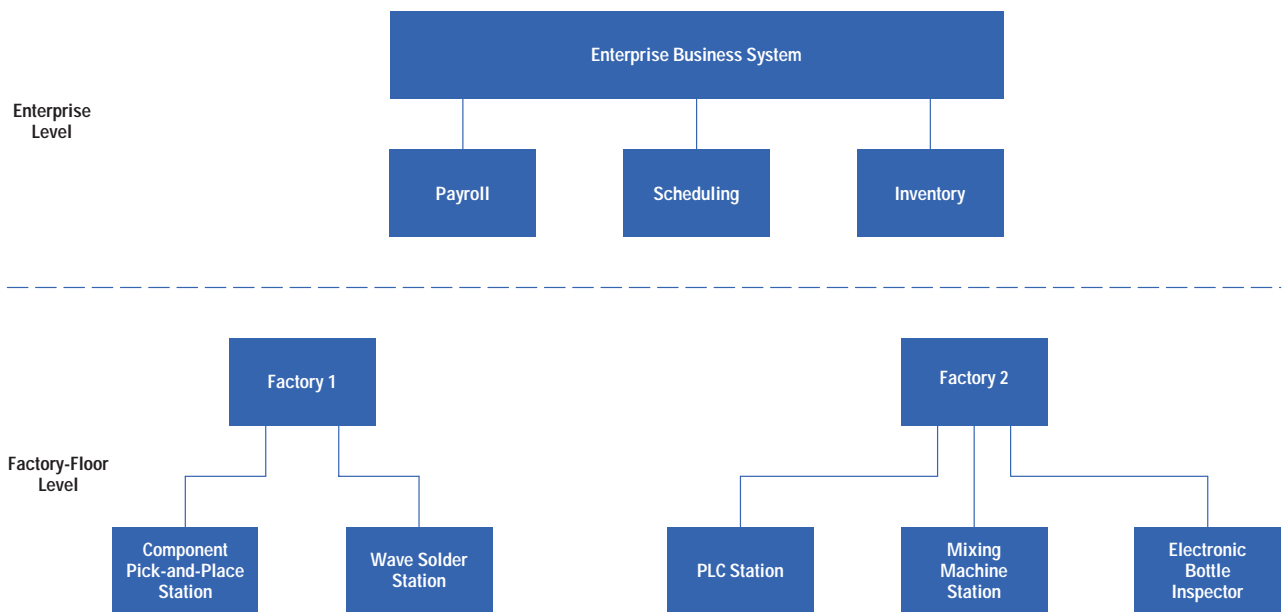
The net effect is that today companies find it difficult and expensive to integrate factory-floor systems with each other and with business software running at the enterprise level. This is unfortunate because the dynamic nature of the marketplace and the desire to reduce inventory levels have made the need for such integration very high.

Marketplace Dynamics

Over the last decade, the marketplace has become increasingly dynamic, forcing businesses to adapt ever more quickly to changing market conditions. Computer systems now experience a continuous stream of modifications and

Figure 1

Computing at the enterprise and factory-floor levels.



upgrades. Generally, this has forced business systems to adopt more real-time behaviors and production systems to become more flexible. It has also increased the frequency and volume of data transferred between business and production systems and between the many production systems.

There has always been a requirement to transfer information between computers in an organization, both horizontally between computers at the same functional level, and vertically between computers at different functional levels. In the past, manual data entry was an often-used approach. Hard-copy printouts generated by business management systems would be provided to operators who manually entered the information into one or more production systems. Although this was an acceptable approach in the past, such an approach is not sufficiently responsive in today's dynamic business environment. As a result, the need for electronic data transfer capability between the various business management and production level computers is now very high.

Electronic Data Transfers

Integrated business software with built-in support for data transfers between components is sometimes used at the business management level. While this minimizes the effort required to exchange data between the various components of enterprise business systems, it is often inflexible and restrictive with regard to what can be exchanged and when exchanges occur.

Organizations that use a variety of business software packages, rather than a single integrated package, have typically developed custom software for electronic data transfers between packages. Unfortunately, marketplace dynamics require custom software to be constantly reworked. This ongoing rework forces companies to either maintain in-house programming expertise or repeatedly hire software consultants to implement needed changes. As a result, custom data transfer software is not only expensive to develop but also costly to maintain—especially if changes must be implemented on short notice.

On the factory floor, software programmers have been employed to develop custom data transfer solutions that allow the different islands of automation to communicate. As previously noted, this approach is difficult to implement and expensive to maintain. In addition, this approach is often inflexible since the resulting software is usually

developed assuming that the configuration of factory-floor systems is largely static.

When new equipment and application software are to be integrated into the overall system, software programmers don't just prepare additional custom software. They must also modify the existing custom software for all applications involved. For this reason, custom software is often avoided, and electronic data transfer capability is frequently confined to transfers between equipment and software supplied by the same manufacturer.

Differences in hardware (and associated operating systems) and differences in the software applications themselves cause numerous application integration problems. Here are a few examples:

- Data from applications running on computers that have proprietary hardware architectures and operating systems is often not usable on other systems.
- Different applications use different data types according to their specific needs.
- Incompatible data structures often result because of the different groupings of data elements by software applications. For example, an element with a common logical definition in two applications may still be stored with two different physical representations.
- Applications written in different languages sometimes interpret their data values differently. For example C and COBOL interpret binary numeric data values differently.

What is needed, therefore, is an off-the-shelf product that is specifically designed to interconnect applications that were not originally designed to work together. That product must automatically, quickly, efficiently, and cost-effectively integrate applications having incompatible programming interfaces at the same or different functional levels of an organization. HP Enterprise Link is such a product.

HP Enterprise Link is an interactive point-and-click software product that is used to connect software applications (such as business planning and execution systems) to control supervisory systems found on the factory floor. HP Enterprise Link greatly reduces the cost and effort required to interconnect such systems while eliminating the need for custom software.

The Data Transfer Problem

The problem of transferring data from one software application to another is conceptually simple: just fetch the data from one system and place it in another. In practice the problem is more complex. The following issues arise when trying to implement electronic data transfer solutions:

- There must be a way to obtain data from the software application serving as the data source. Such access, for example, might be provided by a library of callable C functions.
- There must be a way to forward data to the software application serving as the data destination. For example, data might be placed in messages that are sent to the destination application.
- There must be a specification of exactly what to fetch from the source application and exactly where to place it in the destination application.
- The data being transferred must be translated from the format provided by the data source to the format required by the data destination.
- There must be a specification of the circumstances under which data should be transferred and a way to detect when these circumstances occur.

All of these issues are addressed in HP Enterprise Link.

HP Enterprise Link

HP Enterprise Link product consists of the three components shown in **Figure 2**:

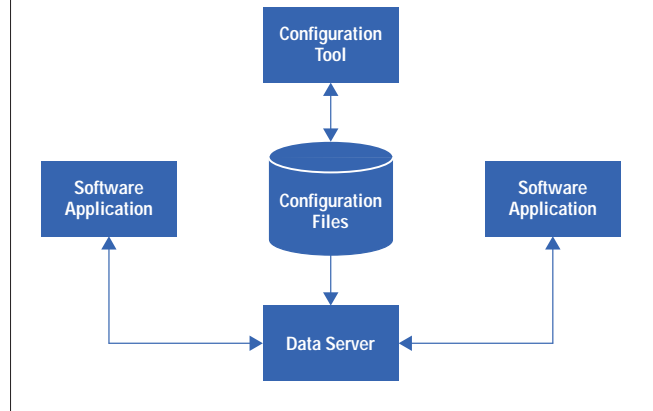
- An interactive configuration tool. This interactive window-based application allows users to direct the movement of data between two software applications.
- A data server. This noninteractive process runs in the background. It moves data in accordance with the directives that the user specified with the configuration tool.
- Configuration files. This is the set of mappings and trigger criteria created by users. The data is stored in configuration files. These files are created and modified by the configuration tool and read by the data server.

Linking Components

The HP Enterprise Link components listed above have the common goal of enabling users to create middleware that

Figure 2

The components of HP Enterprise Link.



maps components with different interfaces together for data transfer.

In HP Enterprise Link, the combination of a single source address and a single destination address is called a *mapping*. A unit of data at the specified source address is said to be mapped to the specified destination address. In other words, it can be read from the specified source address and written to the specified destination address.

Although a mapping deals with the transfer of a single unit of data, real-world situations usually require the transfer of many units of data simultaneously. Therefore, HP Enterprise Link collects mappings into groups called methods. A method contains one or more mappings.

Mappings describe what to transfer and where to transfer it, whereas triggers describe exactly when to do the transfer. Data is actually transferred whenever a specified trigger condition is satisfied. This condition is called the trigger criterion. There are many possible trigger criteria such as:

- Whenever a unit of data at a specified source address changes value
- Whenever a unit of data at a specified source address is set to a specified value
- Whenever the source data becomes available—such as arriving in a message
- At a preset time of the day or a preset day of the week.

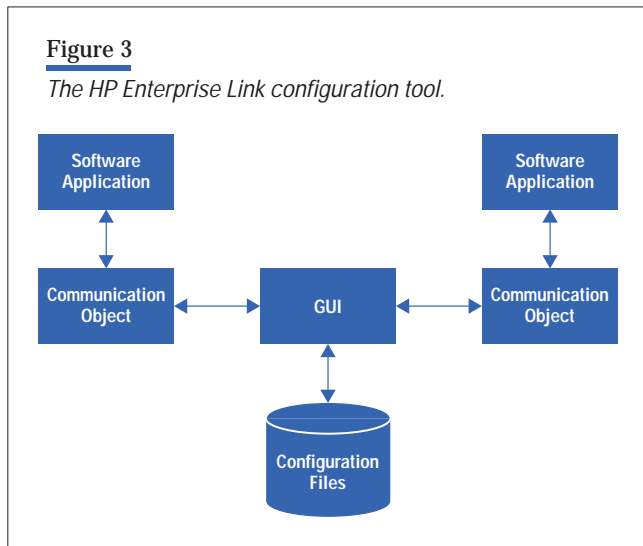
HP Enterprise Link considers trigger criteria to be part of the definition of a method. All the mappings for a single method share the same trigger criteria. Whenever the trigger criteria are met, HP Enterprise Link transfers—in unison—all the data specified by the method's mappings.

Multiple methods can simultaneously exist in HP Enterprise Link. For example, a user can create one method to transfer a particular production recipe from a business enterprise system down to a factory-floor control system. Conversely, raw-material consumption information for the recipe currently in production could be transferred periodically from the factory-floor control system up to the business enterprise system, using a second method.

The Configuration Tool

The HP Enterprise Link configuration tool provides users with a view of each software application's name space, and the tool graphically depicts what data to transfer and under what circumstances such transfers should occur (**Figure 3**).

The HP Enterprise Link configuration tool is composed of communication objects and a graphical user interface (GUI). Communication objects are used to obtain namespace data that is unique to each application and to provide application-specific windows. The configuration tool provides the user with an easy-to-use point-and-click style GUI.



All dependencies on particular software applications are encapsulated in communication objects. The configuration tool's communication objects provide the following functionality:

- They fetch namespace information from communicating software applications for presentation to the user.
- They provide routines to create and manage application dependent control panel widgets, such as those used to specify triggers unique to a particular software application.
- They provide routines to tell the GUI exactly what functionality is supported by a communication object. For example, can the application software serve only as a data source (supply data values), or can it serve as both a data source and a data destination (both supply and use data values)?

There are three important windows in the configuration tool's GUI: the Edit Method window, the Edit Mapping window, and the Trigger Configuration window.

Edit Mapping. The Edit Mapping window is used to create new mappings (**Figure 4**). The namespaces of both the source software application and the destination software application are shown. They are graphically displayed as tree diagrams. This makes it easy for users to specify which data to move where. They don't have to remember the names of data sources or data destinations. Instead they just choose from the displayed list of possibilities. The side-by-side display of application namespaces makes it much easier to integrate the applications.

Tree diagrams are used because they make large namespaces manageable. A linear namespace display was rejected early in the design of HP Enterprise Link because a flat list representation would only be manageable with software applications having a small namespace. Another advantage of tree diagrams is that most users are already familiar with them from file selector windows found in many software applications.

To create a new mapping the user selects an item from the Mapping Source tree diagram and an item from the Mapping Destination tree diagram, and then clicks the Add Mapping button. A new mapping is added to the mapping table displayed on the Edit Method window (**Figure 5**).

Figure 4

The Edit Mapping window.



Multiple static mappings can be created in a single step using branch assignments. This requires that the last component of the source and destination addresses be identical (so that appropriate mappings can be automatically created). Mappings can also be automatically created at the time methods are triggered. This is called dynamic mapping and requires the user to specify algorithms that can select source addresses and transform these addresses to valid destination addresses.

Edit Method. The Edit Method window (**Figure 5**) displays a method's mappings as a two-column table titled Mappings. Source addresses appear in the left column and destination addresses appear in the right. The data server transfers mapped data from source addresses to destination addresses in the same order as the mappings are listed in this table. The Mappings table makes mappings both explicit and intuitive to the user.

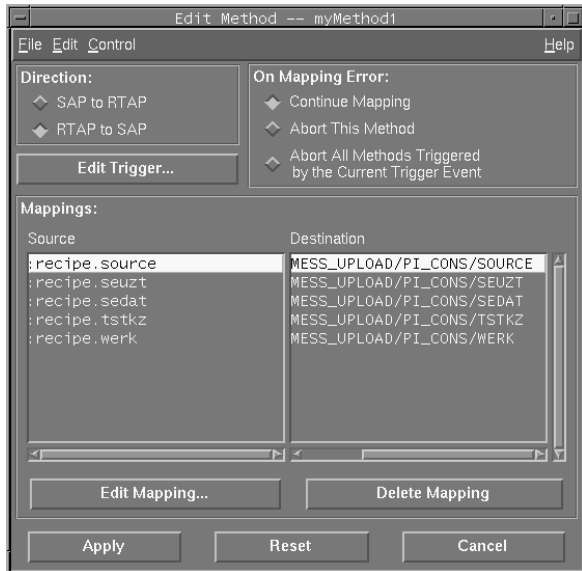
This window allows the user to specify in which direction to transfer data. All of a method's mappings specify data transfers in one direction—from one software application to another. The Edit Method window also allows the user to specify how to respond to errors that occur during data transfers. This will be described later in more detail.

Trigger Configuration. The Trigger Configuration window is used to define trigger criteria (**Figure 6**). This window displays all possible triggers to the user, as well as the currently configured trigger criteria. The Trigger Configuration window is designed to make setting up trigger criteria explicit and intuitive for the user.

The Trigger Configuration window is split into three groups: time triggers, triggers unique to the source application, and triggers unique to the destination application. Time triggers allow the user to specify that data mapping start

Figure 5

The Edit Method window.



at some specified time and repeat at a specified time interval, but be synchronized to a specified hour/minute/second of the day/hour/minute.

Triggers unique to the source application, such as the RTAP (real-time application platform) triggers shown in **Figure 6**, allow data to be mapped when something interesting happens in the source application. For the RTAP triggers in **Figure 6** interesting events include a database value change or the occurrence of an RTAP database alarm. Data can also be mapped when something interesting happens in the destination application.

Thus, triggers allow data transfers to be pushed from the source application, pulled from the destination application, or scheduled by time.

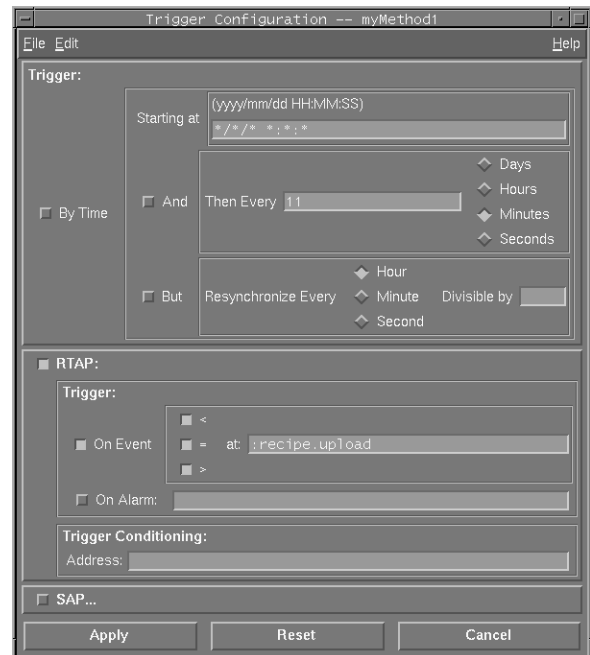
Summary. Using the windows just described, users can create methods with the configuration tool. These methods specify one or more mappings and associated trigger criteria. This information is saved in one or more configuration files. The data server then reads these configuration files to implement the user's methods.

The Data Server

The HP Enterprise Link data server is composed of communication objects, a trigger manager, and a mapping

Figure 6

The Trigger Configuration window.



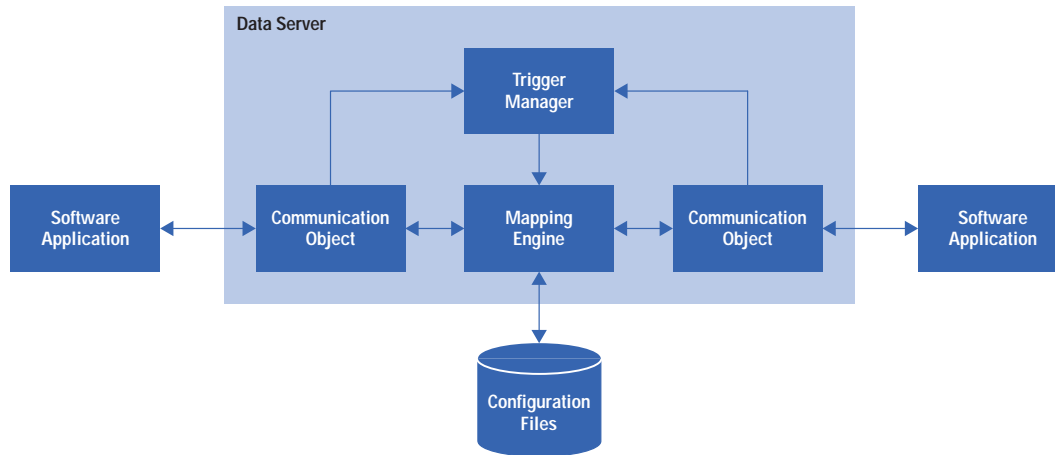
engine (**Figure 7**). Communication objects deal with the problems of generating triggers and getting data into and out of software applications. The trigger manager deals with dispersing Trigger Configuration data, coordinating trigger events, and notifying the mapping engine of trigger events. The mapping engine deals with the problems of reading configuration files, responding to triggers, mapping source addresses to destination addresses, and transforming the data as it is being mapped.

All software-application dependencies are encapsulated in communication objects. Communication objects serve as translators between external software applications and the data server's mapping engine—they translate the software application's native application program interface (API) to the interface used by the mapping engine.

The interface between a communication object and the mapping engine is standardized, with all communication objects using the same interface. For data that is being transferred, the interface consists solely of address-value pairs, where the address is from the application software's namespace, and the value is encoded in a neutral form. Thus a communication object only needs to be

Figure 7

The components of the HP Enterprise Link data server.



aware of its own namespace and how to convert between the software application's proprietary data formats and the neutral HP Enterprise Link data format. For triggers, the interface consists of well-documented interactions between the trigger manager and the communication objects.

Communication objects are usually distributed. They are split into two parts that are interconnected by a communication channel such as a TCP/IP socket. One part of the object is incorporated into the HP Enterprise Link data server process, while the other runs on the same machine as the corresponding software application. When a communication object is not split into two parts, the object, the data server, and the software application must run on the same machine.

Communication objects communicate with their corresponding software applications through whatever mechanism is available. For example, this could be through a serial port, shared memory, shared files, TCP/IP sockets, or an application program interface (API).

When a communication object transfers data, it translates data between the format used by the source software application and the neutral format required by the mapping engine. For example, for numeric values, a communication object may have to translate between binary IEEE-754 floating-point format and the mapping engine's neutral format.

In practice, not all data transfer attempts will be successful. For example, a particular source address might have been deleted, or a destination address may no longer exist. The configuration tool is used to specify what the mapping engine should do in this situation, and the data server must detect the condition and deal with it appropriately. When data transfer attempts fail, the user can have the data server do any one of the following:

- Continue mapping data (ignoring the error)
- Abort all subsequent mappings associated with the current method
- Abort all subsequent mappings and all subsequent methods triggered by the current trigger event (if multiple methods were simultaneously triggered).

The interface between the communication object and the mapping engine is designed to support transaction-oriented data transfers, using commit and rollback. This functionality comes into play when mapping attempts fail. It allows the data server to undo (roll back) all data transfers done in all currently processed mappings associated with the method's current trigger event.

The Running Data Server

When the HP Enterprise Link data server starts up, it reads the configuration files that the user created with the configuration tool. It then prepares to deal with the specified

trigger criteria, usually by notifying the appropriate communication object to detect it. Finally, it enters an event-driven mode, waiting for the trigger criteria of any configured method to be satisfied.

When either a source or destination communication object in the data server detects that a method's trigger criteria have been satisfied, the object informs the data server trigger manager that a method has been triggered. This starts the mapping engine. Alternatively, if the data server trigger manager detects that a method's time-based trigger criteria have been satisfied, the mapping engine starts.

When triggered, the mapping engine requests that the source communication object provide the current data values at the method's configured source addresses. The source communication object obtains these values from the software application, translates the format of all fetched data values to a neutral format, and passes the result to the mapping engine as address-value pairs, with one such pair for each of the method's defined mappings.

The data server mapping engine looks up the destination address that corresponds to each source address. This lookup results in a new list of address-value pairs, with the address now being the destination address, and the value unchanged (and still expressed in the mapping engine's neutral format). To minimize the impact on performance, this lookup is implemented using a hash table.

The mapping engine sends the new list of address-value pairs to the destination communication object. The destination communication object converts the received values into the format required by the destination software application, and writes the converted result to the specified addresses in the destination software application.

Communication Objects and Software Applications

There are two fundamental ways for software applications to provide communication objects access to their data: the *request-reply* method and the *spontaneous-message* method.

In the request-reply method, the communication object sends a software application the address of a wanted data unit in a request and receives its current value in a reply. With this method the communication object controls the data transfer. It determines which unit of data to read and when to read it. Structured Query Language (SQL) and

real-time databases are two examples of software applications that employ the request-reply method.

In the spontaneous-message method, communication objects receive data, usually as messages, from the software application whenever the application chooses to send it. With this method the software application controls the data transfer. It determines which data to provide and when to provide it. SAP's R/3 product is an example of a software application using the spontaneous-message method.

The method that a software application employs to provide external data access determines the trigger criteria that are possible for that application's communication object. The request-reply method allows event, value, and time-based trigger criteria since the communication object controls the data transfer. The spontaneous message method is limited to value-based triggering (essentially filtering) because the software application providing the data controls the data transfer.

Spooling

The HP Enterprise Link data server's communication objects must cope with communication failures. This means that outgoing data must be locally buffered until a communication object verifies that the application software, when acting as a destination, has successfully received it. It also means that incoming data must either be safely transferred through the mapping engine or locally buffered when a communication object accepts data from the source application software.

Spooling is especially important if the source application is separated from the HP Enterprise Link data server by a wide area network (WAN). WANs are considerably less reliable than local area networks, and thus are more likely to lose data.

In a typical HP Enterprise Link installation the data server runs on a machine located near or on the factory floor. Production orders are downloaded from the enterprise level to HP Enterprise Link as soon as they are available. The downloaded data is buffered at the factory until it is needed. Using HP Enterprise Link in this way reduces the probability that the factory would lack unprocessed production orders if the WAN is down for a prolonged period of time.

Buffered data must be preserved even if the HP Enterprise Link host machine is shut down or crashes. To do this, HP Enterprise Link stores buffered data in disk-resident spool files.

The amount of storage used to hold buffered data must be restricted to protect the host computer from failure caused by insufficient resources. HP Enterprise Link can limit the size of spool files by controlling:

- The maximum size of spool storage
- The maximum number of messages buffered
- The age of the oldest message buffered.

The user can set any one or all of these limits, using the HP Enterprise Link configuration tool.

Tracing

HP Enterprise Link allows the data being transferred to be monitored by the user. The monitoring is called *tracing*. Tracing is useful for creating an audit trail of the transferred data and for debugging and testing methods. Tracing does not affect the data being transferred.

The configuration tool is used to enable and disable tracing, but it is the data server that generates trace messages when tracing is enabled.

Data can be traced at a number of different internal locations within the data server (see **Figure 8**). Some of the forms in which trace results can be expressed include:

- Data as received by a data server communication object from a source software application. This trace data is expressed using the source software application's native

data format and includes the source address, the value received or read, and the time of transfer.

- Data as sent by a data server communication object to the destination software application. This trace data is expressed using the destination software application's native data format and includes the destination address, the value sent or written, and the time of transfer.
- Data being mapped by the mapping engine. This trace data is expressed using the data server mapping engine's neutral data format and includes the source address, the destination address, the value transferred, and the time of transfer.

Error messages reported by the mapping engine or by communication objects can also be included in the trace output. This ability ensures that the relative sequencing of data transfer messages and error messages is preserved, which greatly aids the user when trying to troubleshoot mapping problems.

Server Data Flow

HP Enterprise Link allows the flow of data in the data server to be interrupted at a number of different internal points (see **Figure 9**). This is useful for isolating the effects of data mappings during debugging and testing. When an information flow is interrupted, data does not pass the point of interruption; instead, the data is discarded.

The flow of information being transferred from a communication object to a software application can be interrupted. Interrupting the flow here allows the data server

Figure 8

Tracing data that is transferred between applications.

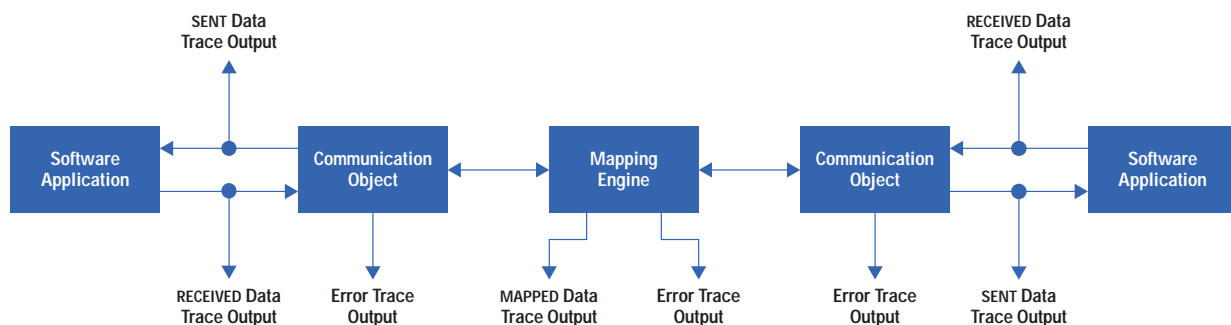
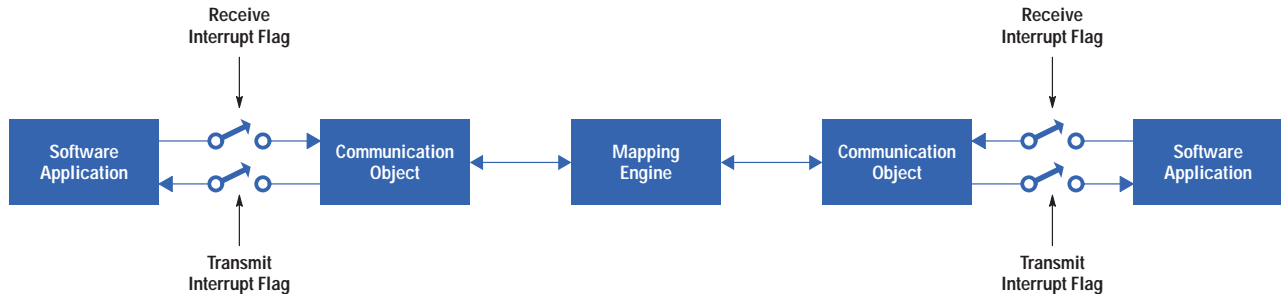


Figure 9

Interrupt locations in the data server.



to read from mapped source addresses, map to new destination addresses, and then discard the data just before it would have been written to the destination software application.

The flow of information being transferred from a software application to a communication object can also be independently interrupted. Interrupting the flow here allows the data server to ignore all data sent to the communication object by the source software application.

Data Integrity

The HP Enterprise Link data server is carefully designed to preserve the integrity of the data being mapped and to map the data exactly once for each trigger event. The design was influenced by considering how to react to communication channel failures and data server process terminations. The circumstances that could cause the data server process to terminate are the following:

- If a person or software process explicitly kills the data server process
- If the host machine suffers a hardware or software failure, loses power, or is manually turned off.

Communication channel failures must be handled carefully. If the communication channel connecting a communication object to its software application fails, the data

being mapped at the time of failure must not be lost or duplicated. Also, after normal operation of the communication channel is restored, communication between the communication object and its application must be automatically established again and all interrupted data transfers restarted.

The following steps are taken to ensure data integrity when communication channels fail:

- For data received from the source software application, the communication object never acknowledges receipt of the data until the data has safely been saved to a disk-resident receive-spool file.
- Data received by the communication object from the source software application is not removed from the receive-spool file until the data has successfully passed through the mapping engine and been forwarded to the communication object responsible for sending it to the destination software application.
- The communication object that sends data to the destination software application only notifies the mapping engine that it successfully received the data after the data has been safely saved to a disk-resident transmit-spool file. Also, it only removes data from the transmit-spool file when the destination software application has acknowledged successful receipt of the data.

