

Systems Architecture Issues in Interfacing BellHawk to your ERP or Accounting System

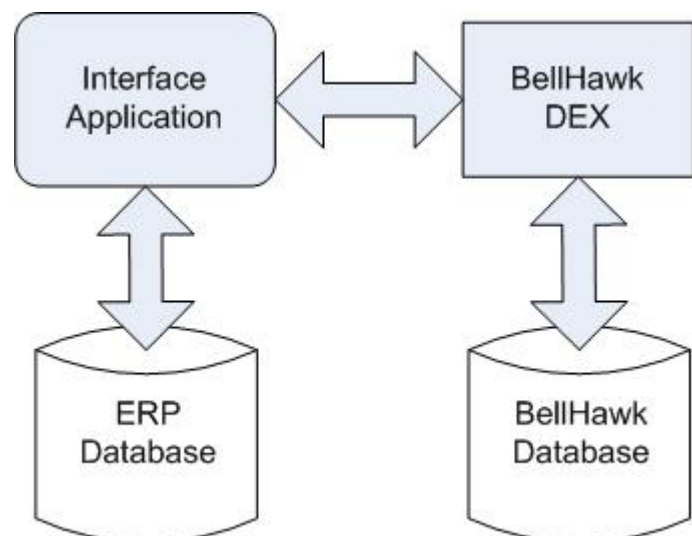
Introduction

This document is intended for use by Information Technology and ERP specialists who need to develop an interface between BellHawk and an Enterprise Resource Planning (ERP), Operations Management, or Accounting system (here collectively referred to as ERP systems). It provides some guidance as to some principles to follow in designing an interface to use the tools provided by the BellHawk Data Exchange (DEX) interface. For details of these tools, please see the DEX User's Manual under the User Manual section of the Product Information link on www.BellHawk.com.

Please note that, because of the large number of ERP systems out there, BellHawk Systems Corporation (BSC) takes the approach of providing tools to facilitate the task of building interfaces rather than attempting to provide a standard interface to any one. For more details on the rationale behind this approach, please see the document "Management Overview of the Issues in interfacing BellHawk to your ERP or Accounting System" under the Technical Papers section of the Product Information link on www.BellHawk.com.

Interface Mechanisms

The most common mechanism that programmers implement is to write an interface application program that interfaces with the BellHawk DEX mechanism and also reads and writes to the ERP system's database. This application may have a user interface to enable the transfer of data objects upon a button push or it may run as a background service transferring data automatically between the ERP system and BellHawk. Such Interface Applications work well and are relatively straight forward to implement when the ERP system has a stable, well documented relational database.



Sometimes the ERP system has a data import and export mechanism of its own, usually in the form of an application programming interface (API). In this case, it is highly beneficial to use this mechanism, rather than reading or writing directly to the ERP database as this isolates the Interface Applications (IA) from changes in the ERP database structure, just like the DEX interface isolates the IA from changes in the BellHawk database.

Sometimes the IA is not developed as a separate application but is embedded within the ERP system, if the developers have access to the ERP source code and are very familiar with its internals. Here we will assume that it is a separate application but the principles are the same.

At the other end of the spectrum we have small accounting systems with proprietary file-based databases. In this case, the best that can be done is to import and export data using comma delimited file (usually Excel .CSV files) in a manual process that usually takes place daily. The BellHawk DEX interface supports the import and export of all its data objects using this file format.

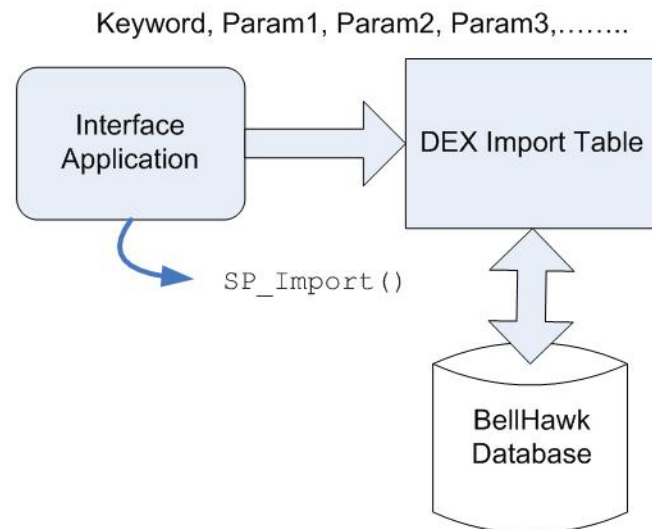
Usually the interface between the Interface Application (IA) and BellHawk/DEX is formed using an ODBC/SQL data link. This enables the IA and the ERP system and BellHawk to exist on different computers, if needed. Some systems, especially older Unix and IBM mainframe systems, do not support ODBC connections. In this case, data is transferred between using comma delimited "flat files" which are exchanged by writing and reading to and from shared file directories. This mechanism is available for BellHawk on request. Basically the ERP system will write the files to a shared directory in an ascii format equivalent to that used in ODBC transfers and BellHawk will periodically scan the directory and pick up the data from new files. A similar mechanism is used for data transfers in the other direction.

The DEX Import Interface

The BellHawk database manages many different types of data objects. We divide these into two types of object, status objects and transaction objects. Status objects describe items like employees and item master parts. Transactions are a result of the scanning done on mobile computers and PCs equipped with barcode scanners.

The Interface Application can export any of the status objects to BellHawk using DEX by writing these status objects to DEX import table. Each entry made in the import table has a Keyword, identifying the object, followed by one or more Parameters identifying and describing the object. Each entry can describe a new object, a change to an existing object, or make a request to specify that the object be marked as deleted. (Note that objects are usually not physically deleted from the BellHawk database to maintain referential integrity in the database. They are just no longer visible).

When the Interface Application (IA) has written all the data objects to the BellHawk database, it then calls a stored procedure `SP_Import()` which causes BellHawk to read in the records in the DEX import table and to put these away in the appropriate table in the BellHawk database. As part of this process, a log table is maintained of all imported objects. Also an error log is generated for any records that have unrecognized keywords or wrong or missing parameters.



The import is based on definitions stored in a DEX import table that is maintained in BellHawk. These definitions can be changed by someone knowledgeable in the internals of BellHawk to add or modify parameters to the import, if needed. These definition tables have setup and edit screens, which are available to BellHawk system administrators, and are a very useful source of information about which parameters are required and which have defaults and what order in which the parameters must be specified.

Frequently, the ERP system does not contain all the parameters needed by BellHawk. This is especially true of data objects such as item master parts records. In this case, the IA can enter the missing parameters as NULL fields and either have the DEX interface fill in a default or leave these to be subsequently entered through the BellHawk setup screens.

Note that certain objects, such as item masters, normally have to be opened in BellHawk and checked as OK before they can be used operationally.

When an object is changed, the IA can export the changed object by designating it as a changed object. In this case, the DEX interface will not change any parameters in the corresponding BellHawk record that is a NULL in the imported data object. Thus it will leave any parameters that are manually setup through BellHawk screens alone.

Some IA programs send the same record over and over again. This is bad practice because of the processing load it imposes on the databases but sometimes the IA programs have no way of knowing which records changed in an ERP database table. BellHawk does do a check for this case and will not update the corresponding record in BellHawk if the new or changed record would result in no changes to the corresponding BellHawk data record.

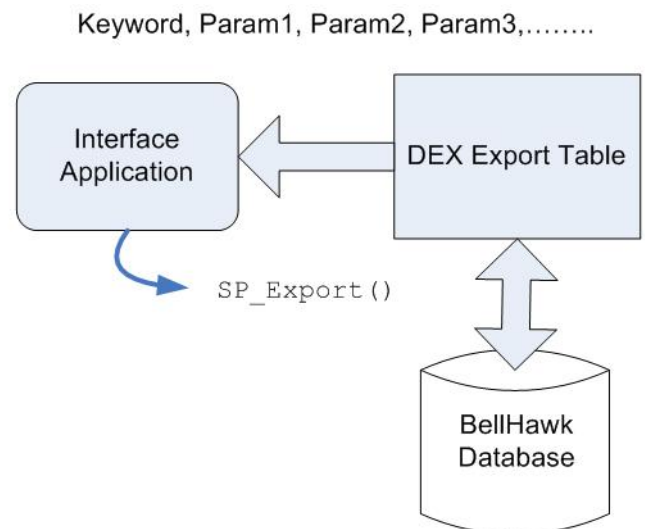
This "no-change" policy is very important as the BellHawk Store and Forward (BSAF) mechanism for communicating with the mobile computers only sends changed records to the mobile computers. Without this policy, if an IA updated all the item master part records, for example, then the changes could clog the communication channels to the mobile computers for minutes or even hours (depending on the number of records) and effectively bring the system to a halt.

A list of all the status objects that can be exported to BellHawk and their parameters are listed in the DEX User's Manual.

The DEX Export Interface

The DEX export mechanism is similar to the import mechanism. In this case the Interface Application (IA) makes a call to the SP_Export() stored procedure. In doing this the IA specified what object it wants exporting and whether it wants all the objects, or just the latest updates or all records within a specified time-date range.

DEX then retrieves these records from the BellHawk database and places them in the DEX Export Table. The IA then reads the records from the interface table and puts the data away in the



ERP database. It then calls another stored procedure to clean up the data records that it has read, as detailed in the DEX manual.

The data is exported in the same Keyword and Parameter format as was used for DEX imports.

In the case of DEX exports, all of the status objects can be exported as well as transaction objects. In the case of transaction objects, the keyword identifies the type of transaction. It also differentiates between similar transactions performed differently, such as performing a "simple receive" transaction as opposed to a "receive against a vendor purchase order" transaction. This provides maximum flexibility to the IA in interpreting the transactional data for the ERP system.

Indirect References and Chaining

In the BellHawk database, as in many ERP systems, many records refer to other records rather than having the data itself stored in the top level record. Thus an order line item may refer to a part. Rather than have the part number stored in the BellHawk order line item table, a reference is stored to the item master parts table.

When BellHawk exports such a record it will "unwind" the references and put the part number, for example, in the export record, rather than put the reference ID in the record. This saves the IA from having to do chained lookups. Similarly on import, DEX expects the IA to have "unwound" the references in the ERP data base and to provide the part number, for example, rather than a reference to the part number. DEX will then take care of looking up the part number and substituting a reference ID when putting the order line item away in BellHawk.

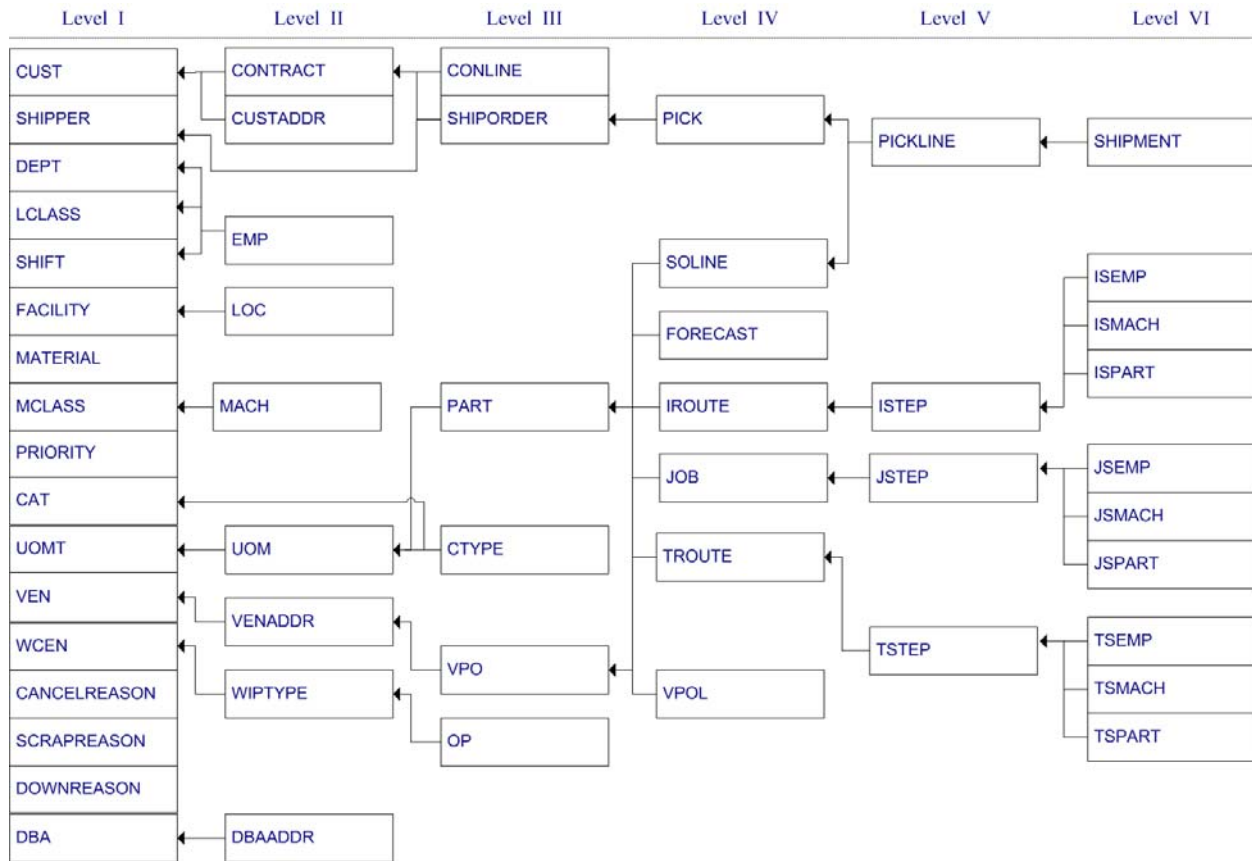
This does require that, for example, the part number is stored in the BellHawk database before the IA exports a line item that references it. Otherwise the import record will be rejected and logged in the import error log table. Usually this requires that the IA has exported the part item master record before the order line item and has exported the required units of measure before that. In some cases, objects such as units of measure are setup manually in BellHawk, as they do not change or they are imported using an Excel spreadsheet as a one-time setup procedure.

The diagram at the top of the following page shows the dependencies of the commonly used status objects on one another. It is important that Level I objects be exported before the Level II objects that depend on them and so forth, unless those objects are manually setup in BellHawk ahead of time.

To assist developers with the chaining issues, DEX does take care of importing Level I objects before Level II objects, and so forth, if they are placed in its import table at the same time. Thus the IA can place a sales order, the sales order line items, the dependent part and customer and ship-to address objects in the DEX import table and DEX will sort out the correct order to import these. It will also ignore the ones it already knows about (if they have not changed).

Please note, however, that DEX will reject a record for which it does not have a dependent record in its database or in the import table as it cannot "unwind" the references in this case.

So the best approach, for objects like sales orders and purchase orders, is to start with a changed order record and export that, then export the line items and then export the dependent objects, unless the IA knows that they have already been exported.



For other objects, like employees and item master parts, it may be more efficient to periodically scan these tables in the ERP system and to look for changes and then export the changed records, either automatically or on a button push on the IA user interface. The most efficient way to achieve this, and to minimize chained records sent, is to modify the ERP tables to have a "time-date last modified" field, if one does not already exist. Then the IA will store the time-date that it last sent that class of object to BellHawk and use this to decide what records to export.

Alternately the IA can store, in its own database table, a checksum or other hash code for each record in each ERP table that relates to BellHawk, along with the time-date last changed, and use this to decide whether the ERP record has changed since it was last sent to BellHawk. This has the advantage of being able to form the checksum over just those fields that are exported to BellHawk and is less dependent on changes to the ERP database.

The object dependency data for BellHawk is stored in DEX tables within BellHawk and can be accessed by an external IA, if needed to assist in making dependency decisions.

Data Master Decisions

An important decision in implementing a BellHawk barcode tracking system is which system is in control of which data objects, BellHawk or the ERP system. This split is usually conditioned by the fact that ERP and accounting systems track money and BellHawk tracks materials as well the actions of people and equipment.

A list of employee's for example may be directly entered into BellHawk or imported into BellHawk from the ERP system. Alternately they may come from a third source, such as a payroll or human resources system. Usually the ERP system holds the Employee master records, which are then exported to BellHawk when changes occur.

Item master part records are usually maintained in the ERP system and exported to BellHawk but it may make more sense to reverse the process, as usually much more data is required in BellHawk for its item master record than is required in the ERP system for accounting purposes.

Bills and routes for making products are usually setup and maintained in BellHawk as it has a richer data representation for how to make items than most ERP system. But these can be imported into BellHawk from an ERP system or manufacturing control system. In either case Jobs or Work orders can be initiated from the ERP system or initiated directly in BellHawk.

BellHawk tracks materials and ERP systems track inventory. These two are joined together because the materials tracked in BellHawk make up the ERP system's inventory. The ERP system may know that it has 300 widgets in inventory. BellHawk will know that it has 100 widgets loose as floor stock and 10 boxes of 20 widgets each in various inventory locations. In this case, BellHawk is always the master and feeds material receipt, consumption, production and shipment records to the ERP system through the IA, such that the ERP system always reflects BellHawk's materials as the total count and value of inventory of each part in stock.

The usual way of handling these inventory changes is to have the IA interact with the tables used by the ERP system's inventory module. In this way, users of the accounting system can see inventory in stock by major location and see its value by item master. Material receipts from vendors recorded by BellHawk can be made to show up as accounts payable. Material shipments to customers also can be made to show up as accounts receivable.

Usually customer orders are entered into the ERP system, or even a separate CRM system, and then exported to BellHawk. But many organizations enter their orders in BellHawk and then export these to the ERP system. The reason for this is that BellHawk can be customized to allow for the capture of specific order data, such as technical specifications, that are unique to a specific organization. In these cases, BellHawk becomes the master as it has far more data than is needed for ERP or accounting purposes.

Similarly vendor purchase orders are usually entered into the ERP system and exported to BellHawk. But, again, there are many exceptions. Many companies find that the data entry screens for entering purchase orders into their ERP systems are too complex for their specific needs. While these data input screens cover every eventuality they can be very cumbersome to use. Many BellHawk users have their BellHawk order entry screens customized to their specific needs, which makes order entry much simpler and less time consuming.

Some organizations use a mixed mode, with supplies for manufacturing being ordered through BellHawk and office supplies ordered through the ERP or accounting system. This works as long as parts in the item master list can be flagged as to which category they belong to.

Some of the choice as to where to initiate jobs and vendor purchase orders depends on whether automated materials requirement planning is being done in the ERP system or in BellHawk. Usually, if the ERP system is being used to do the planning of jobs and raw materials

purchasing, then it makes the most sense for the ERP system to be the master for jobs, BOMs, routes and vendor purchase orders. If BellHawk is being used to do the planning then BellHawk will normally be the master for these data objects.

Some organizations, however, run in mixed mode, whereby they use the BellHawk material planning option to determine the materials requirements and then manually enter purchase orders into their ERP system. These purchase orders are then exported to BellHawk. In this case, BellHawk is usually the repository for bills of material and routes for making parts and is where jobs are initiated.

For some objects, such as sales forecasts, BellHawk is always the slave process, as the sales forecasts are always imported from an external system or from an Excel spreadsheet. For other objects, such as those related to machines, QC, scrap and rework reasons BellHawk is almost always the master.

When it comes to determining the cost that it took to make a product, BellHawk does accumulate the materials as well as the labor and machine time costs for making the product. Usually these elements are exported to the ERP system and then the ERP system decides what costs to apply because it has a much richer set of tools for doing this. It can, for example, use cost tiers to value inventory consumed, which BellHawk knows nothing about as it is a tracking rather than a financial system.

BellHawk can compute the cost of making a product using actual cost for materials received from vendors, plus standard costs for intermediate materials, labor and machine time. The total cost for manufacturing an inventory item can then be reported to the ERP system along with the "material out" transaction; but it is generally better to have the ERP system do the cost calculation as it can use algorithms appropriate to the accounting methods in use rather than BellHawk's relatively simplistic model.

Please note that material, labor and machine time transactions are usually not exported to the ERP system until they have been reviewed and approved by a production manager or supervisor. This is because employees make mistakes in scanning and data entry that can only be detected by a knowledgeable person. After these entries have been corrected and approved then they will be exported to the ERP system.

Please also note that transaction data is sent incrementally as it occurs or is approved, as appropriate and that, for example, multiple labor hour transactions may be received for the same job or work order from BellHawk. The IA needs to take account of this.

So you can see that it is important to make these decisions about which system is master of which data objects before implementing the Interface Application.

Other Issues

An important choice in the design of the interface is the level of automation. At one end of the spectrum, we can simply import and export data using Excel spreadsheets. This requires little investment in initial setup but has a long term penalty in terms of the labor hours to do the exports and imports manually. At the other end of the spectrum are fully automated systems that exchange data in near real-time with little or no human intervention.

One area that these decisions are important is in the area of handling shipping transactions. The IA can simply post the shipment as a decrement in inventory and an increment to accounts payable. Alternately, the act of posting the shipment transaction could automatically print out or FAX or Email the invoice. These are important issues in which the cost of automating the interface needs to be traded off against the labor savings resulting from the automation of certain functions.

The IA developer needs to have an in-depth knowledge of the table structures of the ERP or accounting system or the ERP system's interface API. In addition the developer will probably need between one to three days of consulting support from a member of BellHawk System's technical staff to help plan the design of the IA and to help with mapping issues between the ERP data structures and BellHawk data structures.

Building an Interface Application (IA) is not a trivial job but the features provided by DEX make this a lot easier. This development task should take between a couple of days to two or three weeks of solid work depending on the number of objects to be exported, the complexity of the ERP database or interface API, and the level of automation of the interface.

In summary, the cost of implementing the interface between an ERP or accounting system and BellHawk is usually commensurate with the cost of the BellHawk software licenses. This cost, plus the cost of BellHawk licenses, is usually much less expensive than alternatives such as customizing your ERP system or replacing your current ERP system with a new system.