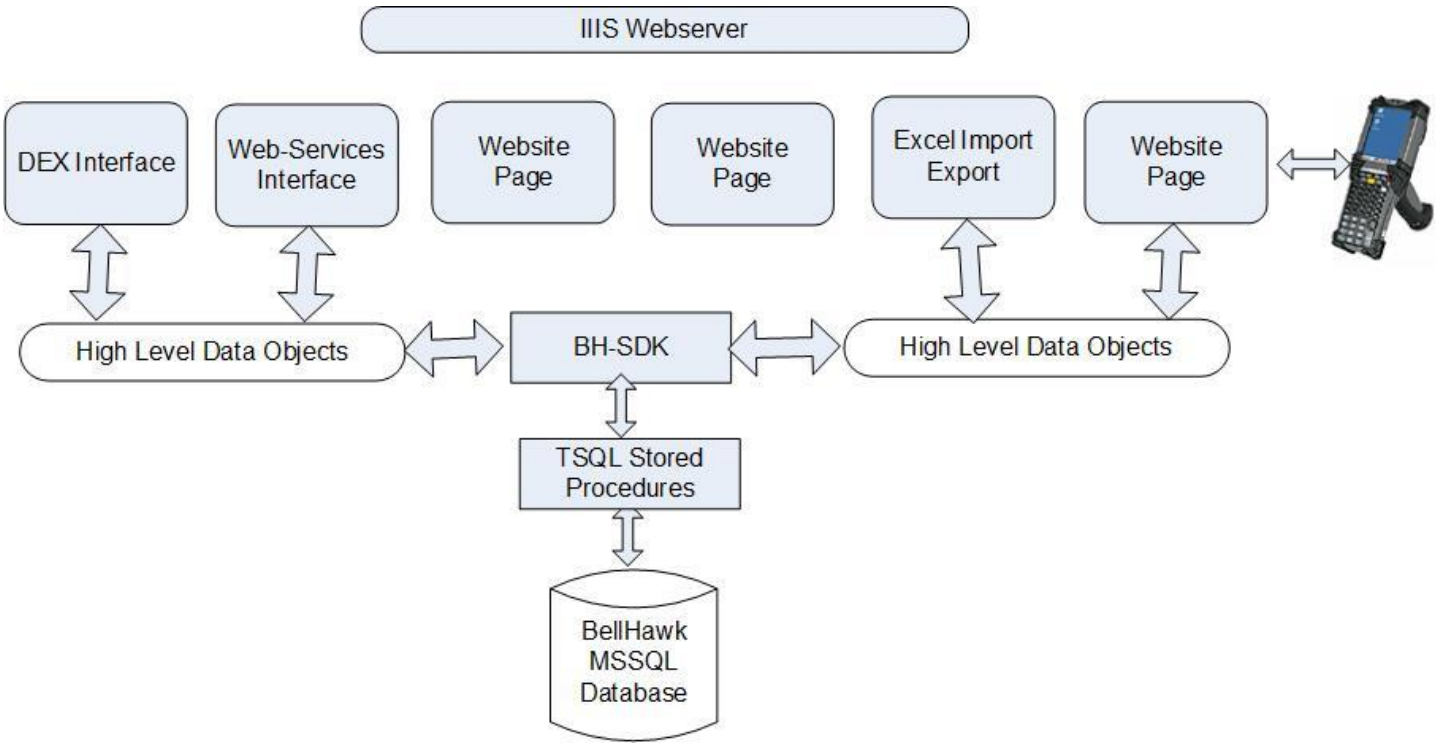
BellHawk Data Sheet
High Level Data Objects (HLDOs)



Introduction

BellHawk uses High Level Data Objects (HLDOs) to communicate with its own database, via the BH-SDK interface to the Tau-Adaptor rules-based engine. They are used by external systems to communicate with BellHawk via its web services interface or DEX interface. They are also used to define the parameters for rules that configure BellHawk data capture, barcode-label-printing, and Excel imports and exports.

Data Exchange

Data exchange with BellHawk can be performed using:

1. The Excel Setup Screens in BellHawk – to import and export BellHawk setup data.
2. The DEXEL Setup screens in BellHawk – Excel import and export for advanced users.
3. The BellHawk BHSDK - .Net Software Development Kit
4. The BellHawk web-services (SOAP/XML) interface

5. MilramX– a framework and toolset for implementing robust 24x7 data exchange interfaces between BellHawk and other systems. This uses high level data objects for both BellHawk and the systems with which it exchanges data

Representation

These data exchanges are all performed in terms of High Level Data Objects (HLDOs), which abstract all the details of the mechanisms needed to lookup, fetch and store data into a universal format that can be exchanged between systems without regard for the communications mechanisms used or how the data is stored in databases or the operating systems used.

An HLDO consists of a keyword, such as “ITEM” and a set of parameter-name:parameter-value pairs of strings, such as “ItemNumber”:”ABC123”. Because parameter values are expressed as strings, they are independent of the data representation used in the underlying databases or interfaces. By convention we use “CamelBack” notation for parameter names, with no spaces (to avoid mistakes).

This is very similar to JSON strings and, in some cases, HLDOs are represented, within BellHawk, in the form of JSON strings.

Each HLDO has an XML metadata representation that describes how the high level data object relates to the schema used by the underlying database or web services interface.

Metadata Representation

This metadata enables:

1. BellHawk to automatically generate the SQL code to translate between its underlying database and the HLDO instance data it imports and exports through its Excel interfaces, its BHSDK, or its web-services interface.
2. MilramX to automatically generate SQL code to translate between underlying databases and HLDOs exchanged between systems.
3. Verifying that the data being imported or exporting is in an acceptable format and does not contain any characters (such as control characters) that would be unacceptable to a target system or to code that is processing the HLDO data.
4. Setting of defaults for unknown values when data is stored into a target system.
5. Providing documentation of how HLDOs relate to underlying database, SDK, or web-services interfaces

The XML metadata for all the HLDOs for a specific system (such as BellHawk) are contained in a single file, such as BellHawk.XML. While XML is supposed to be human readable, on the scale of complex databases such as BellHawk, they can be very difficult to edit or create.

As such, we have a variety of mechanisms for creating and editing the HLDO definitions using Excel spreadsheets, which are explained in the “BellHawk High Level Data Object User Manual”.

Data Types

With HLDOs, and through its user interface, BellHawk makes extensive use of name:value pairs to describe the parameters of data objects.

Every parameter has a specified field data type. These field types are used to check that a data value being set for a parameter is valid; for example that a field type of DATE does indeed contain a string value that can be converted to a date without error. They are also used to check that the value of a parameter entered by a user or being retrieved from an external system or database is valid and does not contain invalid characters for the data type.

The valid data types are:

TEXTID – These are used to specify parameters that are lookup parameters and also text foreign key parameters whose value that must match key values in other tables. These can contain any ASCII character except non printing (control) characters and the percent (%) , comma (,) and double quote (“) characters. Unlike TEXT fields they cannot contain an empty string.

TEXT – Contains any ASCII characters except for non-printing (control) characters. An empty string "" is also a valid TEXT string.

INTEGER – Contains numeric digits. May be prefixed by + or – sign.

FLOAT – Can be prefixed with an optional + or – sign followed by one or more numeric digits. These may be followed by a decimal point and one or more numeric digits. This may be followed by an “e” or “E” for an exponent, followed by an optional + or – sign, followed by one or more numeric digits. An example is 34.79e-12

DECIMAL – May be prefixed by an optional + or – sign, followed by one or more numeric digits, followed optionally by a decimal point and one or more numeric digits. An example is +34.87.

DOLLAR – This is a DECIMAL number that may have an optional $ following the + or – sign. It may also contain commas as separators. The $ sign and commas are stripped out on input and the result is stored in the database in a numeric data format appropriate to the column type in the database. On retrieval the parameter value is returned as a DECIMAL string.

DATE – must be in any valid date format for the system being used. Examples are 10/31/2009 and 2009-10-31.

DATETIME – must be in any valid time data format for the .Net system being used. An example is 10/29/2009 3:35PM.

ID – The ID type is special, in that it is used to identify Auto-Index fields that are generated by SQL Server. These fields can be read from the database but will automatically be excluded from being written when a data object is stored back in the database. These fields will automatically be generated by SQL on a SQL INSERT and reused on an UPDATE.

NUMID – These are used to specify the field types for fields that contain foreign key references to auto ID fields in other databases.

MLTEXT – Multi-Line Text. Same rules as for text except these can contain embedded end of line characters.

YNBOOL – Translate to “Y” or “N” in Excel spreadsheets but are stored in the database as 1 or 0. They can be used to specify the 0/1 no/yes data types found throughout BellHawk.

UDP – User Defined Parameter string in JSON format, such as {“ItemNumber”:”ABC123”;”Description”:”Big Widget”}

HLDO Benefits

The big benefit of HLDOs is that they are independent of the underlying system’s architecture or data representation, enabling data transfer between Windows, Linux, Apple, and Android and even legacy IBM worlds without needing to know about the underlying data representations used by different computers.

HLDOs are also human readable in their various formats enabling data analysts and end users to easily read and understand these data objects.

