# The HBA Language

# 1. Introduction

The HBA language is designed to help users to convert the text records into HTML table format. The syntax of HBA language is simple. With several lines of HBA codes, the users are able to create a report in HTML format.

The HBA language is easy to use. With HBA language, the users don't have to know HTML language to generate reports in HTML table format.

The language is named after its typical usage – to generate HTML report for Fibre Channel HBA – Host Bus adapter. To implement an enterprise SAN ( Storage Area Network ) solution, an important step is to identify the driver and firmware versions of FC HBA, and make sure they are on the support matrix.

Storage vendors such as EMC and IBM usually publish support matrix, on which lists all the supported driver and firmware versions. The listed driver/firmware versions have been thoroughly tested and therefore fully supported by the storage vendors. To avoid un-necessary troubleshooting, the first step for SAN implementation is usually to make sure the driver and firmware versions are on the verdor's support matrix. The HBA language will help field engineer to create HTML report of HBA very easily.

## 1.1 Features

### Text Parsing

HBA allows a user to search through a set of data using the functions provided. The user will be able to state the record name in the command input file.

### HTML Handling

HBA has the ability to describe HTML output clearly, and handle the HTML tags.

### Data Comparison

HBA allows a user to compare the value of text string in the text file with the pre-defined string. If the actual string is different form the pre-defined string, the user can specify the actions, usually to append extra HTML table cells to HBA report.

## 1.2 HBA Sample Syntax

### Text parsing

```
//sample HBA syntax
START;                  //start of the program
INFILE = “….” ;         //text file name
OUTFILE =”…” ;          //output html file name
DELIMITER = “...”;   //the delimiter to separate the text records
```

**Data Comparison**

```
//HBA code
IF NAME = "…" THEN VALUE = "…"; //expected value for specific record
```

HBA language will search for the record in the text file and compare the value with expected value. Related HTML output will be generated.

# 2. Tutorial

## 2.1 An example

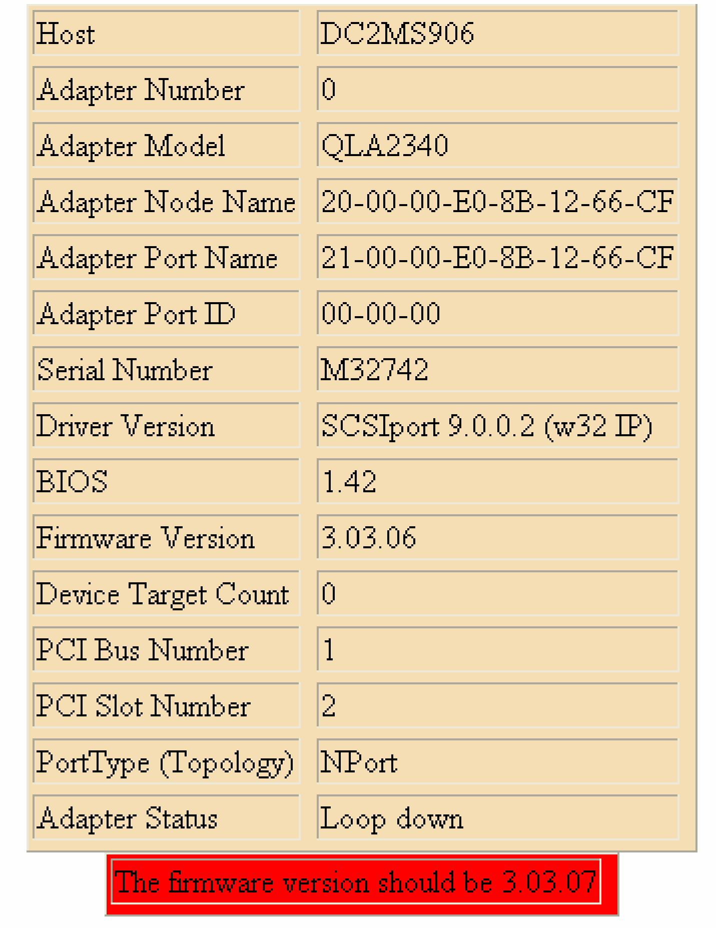
A sample program file “sample.hba” is provided for tutorial purpose.

```
//sample.hba
//start of the program

START;

// input file in text format

INFILE = "text.txt";

// output file in HTML format, which is converted from input text file

outfile = "output.html";

// delimiter for the records in the text input file

delimiter = ":";

// the name of the record which HBA language should search in input
// text file; and the expected value. In this example, HBA will look for
// a record which contains string "Firmware" and compare the related firmware
// version with the "3.03.06". Keywods could be lower-case as well.

IF NAME = "Firmware" THEN VALUE = "3.03.07";
```

The sample program converts “text.txt” into “output.html”, which is in HTML table format.

```
//text.txt

Host                    :  DC2MS906
Adapter Number            :  0
Adapter Model            :  QLA2340
Adapter Node Name         :  20-00-00-E0-8B-12-66-CF
Adapter Port Name         :  21-00-00-E0-8B-12-66-CF
Adapter Port ID          :  00-00-00
Serial Number            :  M32742
Driver Version           :  SCSIport 9.0.0.2 (w32 IP)
BIOS                    :  1.42
Firmware Version          :  3.03.06
Device Target Count       :  0
PCI Bus Number            :  1
PCI Slot Number           :  2
PortType (Topology)       :  NPort
Adapter Status           :  Loop down
```

The generated HTML table will be :



| | |
|---|---|
| Host | DC2MS906 |
| Adapter Number | 0 |
| Adapter Model | QLA2340 |
| Adapter Node Name | 20-00-00-E0-8B-12-66-CF |
| Adapter Port Name | 21-00-00-E0-8B-12-66-CF |
| Adapter Port ID | 00-00-00 |
| Serial Number | M32742 |
| Driver Version | SCSIport 9.0.0.2 (w32 IP) |
| BIOS | 1.42 |
| Firmware Version | 3.03.06 |
| Device Target Count | 0 |
| PCI Bus Number | 1 |
| PCI Slot Number | 2 |
| PortType (Topology) | NPort |
| Adapter Status | Loop down |

The firmware version should be 3.03.07

The following are the steps how to run “sample.hba” in a HBA environment:

1) open a DOS prompt, go to the directory where smple.hba resides
2) run “java HBA sample.hba”
3) you will find the output.html file under the current directory

## 2.2 Installation steps

HBA language is developed for Windows environment. You need the followings software to run HBA language properly.

Windows 2000/XP
JAVA J2SE v1.4.2 or higher
HBA language package

HBA language installation steps:

a) unzip the hba.zip to C:\HBA directory
b) on DOS prompt, run “set classpath=%classpath%;C:\HBA”

# 3. Language Reference Manual

## 3.1 Lexical conventions

### 3.1.1 Comments

A Java style comment is supported. The characters // introduce a comment, which terminates with the new line. // has no special meaning inside comment line.

### 3.1.2 Whitespace

Whitespace is defined as the ASCII space, horizontal tab.

### 3.1.3. Identifiers

An identifier is a sequence of letters and digits; the first character must be alphabetic. Upper and lower case letters are considered different. Identifier -> letter ( letter | digit | ‘_’ )*.

### 3.1.4. Keywords

The following identifiers are reserved for use as keywords, and may not be used otherwise:

INFILE
OUTFILE
START
IF
THEN
NAME
VALUE
DELIMITER

### 3.1.5 Numbers

A number consists of digits.

### 3.1.6 Strings

A string is a sequence of characters enclosed by double quotes ‘ " ‘. A double quote inside the string is represented by two consecutive double quotes.

### 3.1.7 Separators

The following ASCII characters are separators:

(<br> )<br>;

### 3.1.8. Built-in variables

The following identifiers are reserved for use as built-in variables, and may not be used otherwise:

| | |
|---|---|
| Fin | Command input file name |
| Fout | HTML format file name generated by HBA |
| DeLi | Delimiter specified by users |
| Fnm | Specific field name |
| Fvlu | Expected field value |

## 3.2 Expressions

Primary expressions include identifiers and constants.

### 3.2.1 Identifier

An identifier is a primary expression.
.

### 3.2.2 Constant

A constant is a right-value expression, which will be evaluated to the constant itself.

## 3.3 Operators

#### 3.3.1 Assignment operators

variable = expression

The = (assign to) operator groups right-to-left. The value of the expression replaces the object referred to by the variable. The operands need to have the string constant type.

## 3.4 Statements

Statements are executed in sequence.

### 3.4.1 Expression statement

Statements are in the form of expression.

### 3.4.2 Conditional statement

if ( expression ) then statement

The expression is evaluated first, and if it is non-zero, the statement is executed, which is a expression as well.