

# The Jbol Migration Path for migrate!COBOL Conversions



# CONTENTS

|  |          |
|--|----------|
| <b>The Business Case .....</b>                         | <b>3</b> |
| Executive Summary.....                                 | 3        |
| The problem.....                                       | 3        |
| The solution as offered by MSS.....                    | 3        |
| <b>The Technical Solution.....</b>                     | <b>5</b> |
| Batch Infrastructure.....                              | 6        |
| Database Access.....                                   | 6        |
| Online Infrastructure.....                             | 7        |
| J2EE Environment.....                                  | 7        |
| TP Monitor Environment .....                           | 8        |
| XML communication.....                                 | 8        |
| Screen Features.....                                   | 9        |
| The Admin Console.....                                 | 10       |
| Error Handling.....                                    | 13       |
| webManager.....  | 14       |
| Compatibility.....                                     | 15       |
| Deployment.....  | 15       |
| Onward Development/Refactoring of the Application..... | 15       |
| Reliability/Scalability.....                           | 16       |

# The Business Case

## Executive Summary

Much has been said lately about the age of the Cobol programmer and the coming worldwide staffing crisis. This paper looks at how this problem will impact businesses that run Cobol based applications and offers a solution that will eliminate the problem entirely and open legacy applications up to modernisation.

No major organization is currently offering a viable solution to the problem. MSS is the first in this field. Initially we target Unisys mainframe Cobol. Our subsequent plans are to address the largest market segment, core applications running on IBM mainframes.

## The problem

There are about 200 billion lines of Cobol code in the world and it is said that this is increasing at a rate of 5bn lines per year. Many of these applications have their origins in the 1960 when Cobol was first used in mainframe applications. There is a lively debate going on about the future of Cobol.

A recent article sited in the Financial Times stated that half of mainframe and Cobol programmers are over 50 years old, i.e. the average programmer is 57. This means that in five to ten years there is likely to be an increasing shortage in Cobol programmers that could impact business in a number of ways:

- Increase in the cost brought about by skills shortage
- Longer time to market for new products that require Cobol skills
- Difficulty in maintaining Cobol based applications especially where changes in legislation and compliance demand changes in business processes
- The low number of young people willing to be trained in Cobol programming will make this a growing problem for companies that use this language.

This will initially impact on the lesser-known Cobol dialects such as Unisys and VAX who already face a skills shortage because of the relative obscurity of the operating environment. Larger corporations may find that they can push the problem out but there are many smaller and medium sized companies that use mainframe computer who will experience the problem first, if not already. Several mainframe users we have spoken to have already sited this is a strategic issue for their businesses.

## The solution as offered by MSS

MSS's migrate!COBOL tool now offers JAVA as the destination language. Development work on this product for the Unisys Clearpath/Libra (MCP Based) market has now been completed and the first customer migrated successfully. The resulting code is as maintainable as the Cobol code was and in fact is mostly a line for line conversion of the original Cobol application. The end user of the application will notice very little difference

when working on the application and will require very little training. Most if not all Unisys mainframe sites run Cobol applications.

MSS's Cobol to Java migration tool provides these clients with four essential benefits:

- True modernisation of the legacy application including replacement of terminal emulation with native web browser access
- Access to a young and enthusiastic work force that insist on developing in modern language
- A massive increase in development productivity allowing the business to react faster to market trends
- A massive reduction in the running costs of their core business applications.
- Carrying forward the vast investments made in the legacy application

# The Technical Solution

The technical challenge is to create a solution that is efficient, compact and maintainable. Of the tools on the market that claim to be able to convert Cobol to Java none comes near to the MSS solution in terms of maintainability and some impose a heavy penalty on the programmer – up to 100 source files per program and 10x the original number of lines of code need to be maintained.

The experience gained from creating the MSS Linc to Java migration tool shows that this is not an inevitable consequence. In fact Linc is a Cobol-like language and the solutions to the most difficult problems have already been developed and proven. Examples are:

- **Data declarations.** Cobol record structures are, effectively, groups of characters that can be grouped together in different ways and manipulated at different levels. For instance a date can be viewed as a whole (e.g. yyyyymmdd) or as a group of three elements (year, month and day). Any of the elements can be changed the date group itself is changed. A migrated program will include the declarations as calls on an addgroup method that builds the internal structure that will mimic what Cobol does. Notice that the programmer doesn't know or care how his data are stored as long as the normal Cobol rules are followed. Also, because the internal structure is hidden, this lends itself to approaching different dialects of Cobol – IBM Cobol has different rules on how signs are stored with numeric data but this can be taken care of by a simple configuration change.
- **Paragraph entry and GOTO constructs.** Cobol allows a simple explicit 'go to paragraph-name' syntax as well as the more complicated 'go to depending' construct. An implicit goto occurs when one paragraph finishes and another starts within a section or a perform range. Jbol treats each paragraph as a procedure which can be called from a controlling structure – the simplest one being a list of all the paragraphs in a section called one after the other. This concept allows a flexible approach at the same time as presenting the programmer with a source structure with which he is already familiar.
- **Database Access.** In Unisys Cobol data is commonly retrieved from the proprietary hierarchical CODASYL based dbms, DMSII, in a positional manner (FIND NEXT) using language extensions built in to the compiler. This is translated into relational terms by extensive use of cursors and cursor routines. The original behaviour is emulated by these routines which are internally optimised to avoid the potential inherent inefficiency of simulating positional access with cursors that may, in relational terms, span a large number of rows.

Essential Features of the new programs that are generated are:

1. A Cobol program becomes a Java class
2. A Cobol section becomes a Java sub-class
3. A Cobol paragraph becomes a Java method
4. A Cobol data item becomes a group class
5. There is a virtual one-to-one relationship between the original Cobol and the generated Java
6. Generated Java code can be mixed freely with new Java code
7. All comments are retained
8. Correct and consistent indentation is provided throughout

The last two points are part of the emphasis we place on maintainability. COBOL programmers very quickly become familiar with the generated code because the layout is so similar.

## Batch Infrastructure

Batch programs are generated as POJOs (plain old java objects). Each one runs in its own JVM (Java Virtual Machine).

Programs run against relational versions of the original DMSII database created (and populated) by MSS's migrate!DATA tool. All mainstream relational databases (Oracle, DB2, SQLServer) are already supported. Others (e.g. MYSQL, Sybase) can be addressed as the need arises.

## Database Access

Database access is performed via dynamically generated cursors, using a simple Java class definition for each table. The calls emulate the original DMSII structures without imposing additional load on the programmer.

A FIND NEXT loop is replaced by a next() call to the named table class. Here is a sample class definition for a DMSII table called ADDR which has been implemented as an Oracle table:

```
public class ADDR extends Dasdl {
    public Var lastname = new Var(Var.CHAR, 20).set("");
    public Var firstname = new Var(Var.CHAR, 20).set("");
    public class ADDR_DS1 extends DasdlSet {
        public String lastname;
        public String firstname;
    }
    public ADDR_DS1 addr_ds1 = (ADDR_DS1)new ADDR_DS1().init(this);
    etc.
}
```

This is used in the program as follows:

```
//Define an instance of the dasdl structure:
public ADDR addr = (ADDR)new ADDR().init(this, databasename);

//emulation of the FIND NEXT loop:
while(addr.findNext()) {
    //process the data. Each field accessible as:
    addr.lastname;
}
```

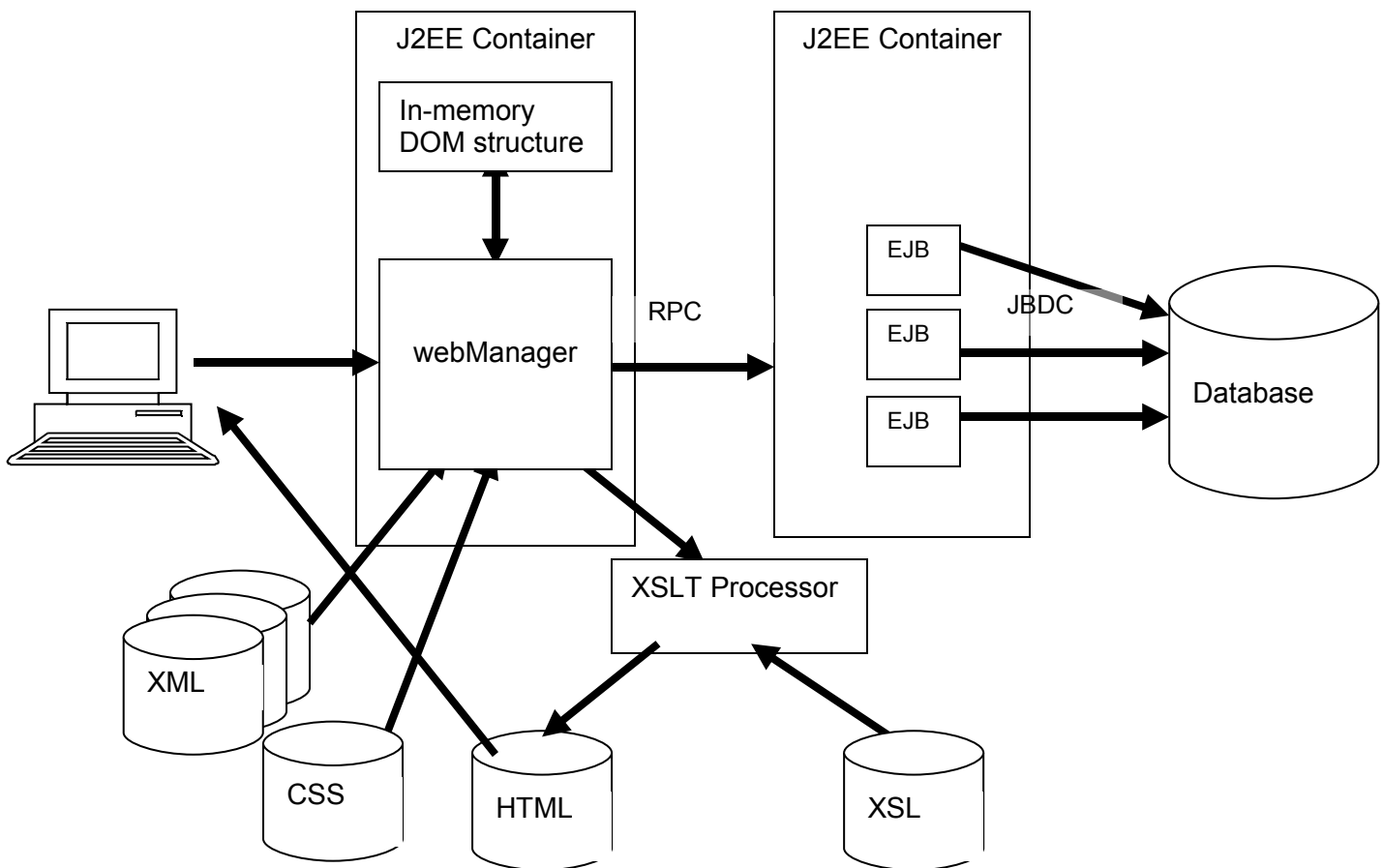
## Online Infrastructure

Please note that the online migration is under development and a final version is not yet available.

Two alternative migration paths are offered – a 'pure' J2EE and a TP Monitor based environment. In both cases the front end will be browser based and follow the MVC (Model View Controller) paradigm.

## J2EE Environment

The standard J2EE model as represented by the following diagram:

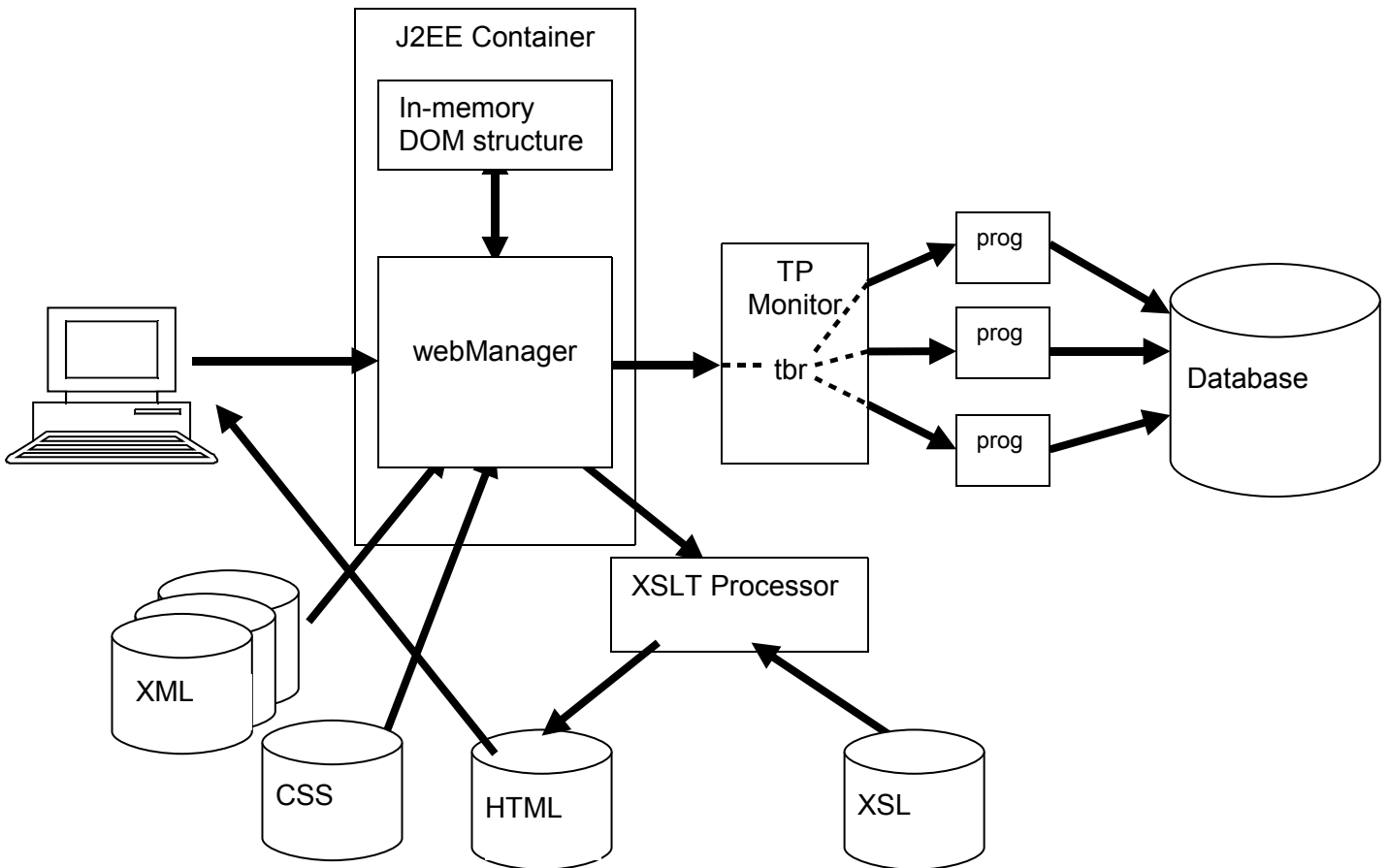


The EJBs contain transaction processing code (several different transaction types per module may be supported).

Look and feel of the forms is controlled centrally by stylesheet processing and all forms can be modified (with immediate effect) by changing the supplied CSS and XSL files (see below).

## TP Monitor Environment

The standard model for an online program in an MCP environment is that of a server handling several (possibly several hundred) different transaction types in a stateless manner. This has the advantage of reducing program start-up time which, in a complex stack-oriented architecture, can become large. But it can lead to very large and unwieldy programs. In this situation MSS would propose a TP monitor/server model something like the following:



The TP Monitor can be MSS's MCS!Lite which has the advantage of being modeled on Unisys COMS and provides features such as agenda processing which may be vital for an installation. Alternatively another message control system (Tuxedo, CICS, IMS) can be accommodated. Any of these will support TBR (Transaction Based Routing) and be capable of controlling programs in multiple copies.

## XML communication

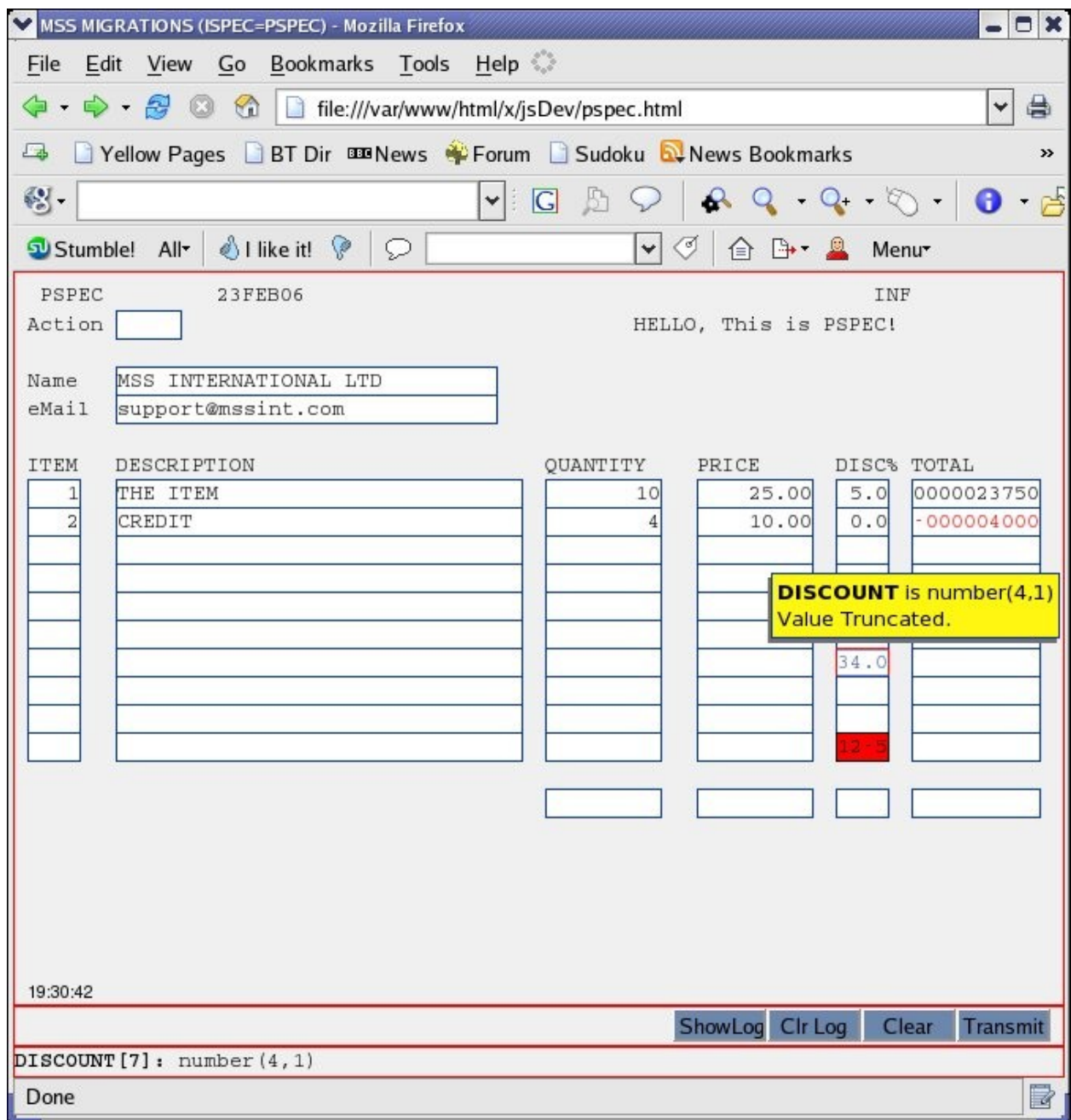
XML files are generated to define the data layouts of the screens (one file per screen). The XML files are processed under the control of a single XSL file to create standard HTML which is sent to the browser. Included in the HTML are JavaScript routines to perform basic validation and style information utilising a CSS file. The CSS file can be modified by any standard tool to change the look of the screens globally so that no individual changes to forms are necessary. Similarly the XSL file can be changed to change the "shape" of the screen (e.g. mapping of column/row to pixels, adding GUI widgets).

An automated process supports the generation of screen formats from SDF. Others will be implemented as required. MSS is aware that there is a variety of screen handling methods in use and has acquired familiarity with most of them during the course of Cobol to Cobol migrations.

## Screen Features

The jBol screen interface is very much more flexible and capable than mainframe interfaces based on T27 terminal emulation (as the great majority of Unisys Cobol legacy applications still are).

The screen below is generated from a mainframe T27 screen (note that this is a Linc screen – Cobol screens are functionally identical):



PSPEC 23FEB06 INF  
Action HELLO, This is PSPEC!

Name MSS INTERNATIONAL LTD  
eMail support@mssint.com

| ITEM | DESCRIPTION | QUANTITY | PRICE | DISC% | TOTAL      |
|------|-------------|----------|-------|-------|------------|
| 1    | THE ITEM    | 10       | 25.00 | 5.0   | 0000023750 |
| 2    | CREDIT      | 4        | 10.00 | 0.0   | -000004000 |
|      |             |          |       | 34.0  |            |
|      |             |          |       |       |            |
|      |             |          |       |       |            |
|      |             |          |       |       |            |
|      |             |          |       |       |            |
|      |             |          |       |       |            |
|      |             |          |       |       |            |
|      |             |          |       |       |            |

19:30:42

ShowLog Clr Log Clear Transmit

DISCOUNT [7] : number (4, 1)

Done

**DISCOUNT is number(4,1)  
Value Truncated.**

This illustrates the following features:

1. Optional "field information" bar which indicates type of field as you move through the form.
2. Optional clickable "mouse button" bar in addition to transmit keys, etc.
3. Automatically filled in decimals (PRICE and DISCOUNT) as you type.
4. Automatic case conversion during input. Fields may be configured to accept lowercase characters (e.g. the eMail field above).
5. Automatically filled "as you type" leading zero's in configured fields (e.g. TOTAL).
6. Automatic "implied decimal" as you type (e.g. TOTAL)
7. Optional negative numbers in red feature.
8. Pop up warnings and error messages occur as you type. The messages are unobtrusive in that they don't require clicks or additional key depressions. They appear and disappear as you move through the fields. Fields in error will prevent transmission of a form.
9. Fields in error are highlighted in red and fields with warnings in a blue colour. This particular warning field indicates that after automatically inserting the required decimals the value was truncated.
10. Optional clock (which can be positioned in any corner).
11. Current field is highlighted (border and background colour change).

In addition to the features visible in the screenshot there are a variety of others:

12. Optional automatic screen re-sizing. The browser window can be set so that it's always the correct size, with no scrollbars.
13. Optional "move to next field" when typing fills current field.
14. Optional debug information window for developers.
15. Optional real-time statistics panel (shows transaction time and other relevant information).
16. Automatic calculation for arrow key navigation. i.e. up and down arrow keys will move the cursor to the field you intuitively believe it should.
18. Fully configurable transmit, reset, help keys.
19. Fully configurable function keys. i.e. If your application uses function keys to input certain data into a particular field and then perform some action, this can be done. Note that only the Firefox and Microsoft Internet Explorer browsers fully support this feature.
20. Input error checking as you type. If a field is unsigned, it won't allow input of negative numbers, etc. This feature only works in Firefox and Internet explorer. Other browsers do not allow the prevention of input characters.
21. Entire form definition is in XML.
22. Multiple scheme support for CSS file defined schemes.
23. Optional user configurable colour schemes.
24. System level configurable defaults include options which allow or disallow users from configuring their own options. Options may be individually allowed or disallowed.

If allowed, the additional information panels (button bar, debug window, field information bar and statistics bar) may be toggled on and off at will. The system administrator can prevent or force users from either toggling or viewing any of the bars.

## The Admin Console

The jBol migration path uses a full-featured front-end system that is very flexible and has many variations. To assist in configuring these options, MSS provide an administration console as part of the product. The console uses AJAX techniques in order to present

required information on demand. For example, if the **Security manager->LDAP manager** option is selected, additional fields will appear which are related to the LDAP option.

The admin console deals with System Parameters and Application Parameters as well as providing a convenient place to handle function key mapping.

To access the console, simply press F5 while logged on as an administrator.

The **System Parameters** Screen appears thus:

The global properties loader specifies how often the application should refresh the properties. There will be some delay – otherwise the overhead could be high – but changes made in the admin console will be reflected in the operation of the application in a short time without having to restart.

The security layer is, perhaps, the most important feature of the console. Mainframe users are used to having few tools to integrate with enterprise security and typically rely on a home-grown password protection system embedded in the application itself. In the meantime, the enterprise itself has standardized on LDAP or Active Directory for general user access. The security layer controls allow an administrator to specify which (of several common choices) flavour of security to use and, at a detail level, enter the required parameters. This, for most Cobol applications is a major step forward in securing a vital resource.

The machine name binding section allows an administrator to describe how he wants the names of stations to appear to the application. Often the station name is used within the application for purposes such as an extra check on access to sensitive data, i.e. terminals in

insecure locations should be denied access. The machine name binding allows this to be carried forward in the new environment.

The personalization section simply allows an administrator to let individual users personalize their own machine.

The **Application Parameters** screen is presented thus:

**MSS Application parameters**

System parameters | **Application parameters** | Key bindings | Log off

**Company parameters**

|                  |   |   |
|------------------|---|---|
| Company name     | <input type="text"/>                                | Specify the company name.   |
| Application name | <input type="text" value="Gsys"/>                   | Name of the application.  |
| Support message  | <input type="text" value="Please contact Support"/> | Message to show in case of the system error.                                      |
| Support e-mail   | <input type="text"/>                                | Setting this value will allow users to send e-mails containing exception details. |

**Server parameters**

|                        |  |  |
|------------------------|--|--|
| Application datasource | <input type="text" value="jdbc/ftDS"/>   | Datasource the application will use.   |
| Logon form             | <input type="text" value="GBMF"/>  | XML form that should be used when starting the application                                   |
| Character set          | <input type="text"/>   | This is the charset used to interpret communication between the application and the browser. |
| CSS                    | <input type="text" value="std.css"/>   | Application CSS - can be customized  |
| XSL                    | <input type="text" value="std.xsl"/>   | Application XSL - can be customized  |
| JSP View               | <input type="text" value="std.jsp"/>   | Application JSP - can be customized  |
| Session timeout        | <input type="text" value="-1"/>  | Time in seconds to remember the session.<br>-1 = forever                                     |
| Action for BYE         | <input checked="" type="radio"/> log off from the application<br><input type="radio"/> run an XML form | Action that will be taken when BYE is called by the application                              |
| Show function keys     | <input type="radio"/> no<br><input checked="" type="radio"/> yes                                       | Should function key labels be shown at the bottom  |

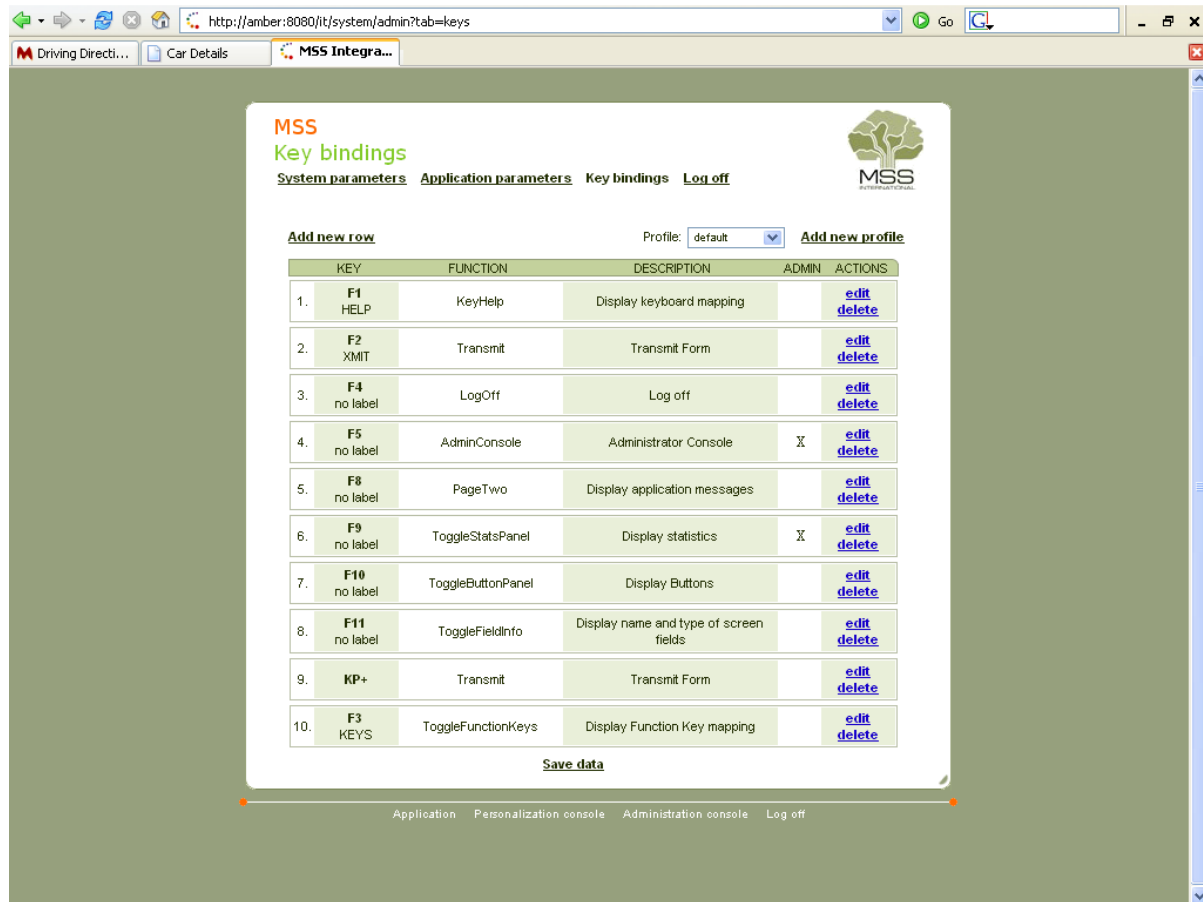
[Save data](#)

Application | Personalization console | Administration console | Log off

The Company Parameters section allows the administrator to document the application's business environment and customize it for different companies (or divisions within a company). In the event of an application error, an error page is displayed along with the Support message and other useful information.

The Server Parameters section configures the application in terms of its look and feel (customizable CSS/XSL/JSP), the data source, and the way it functions for a user (the logon screen and the logoff action).

The **Key Bindings** screen is presented thus:



This screen allows the administrator to customize the key bindings. Key binding profiles may be created and selected within the xml for a particular screen. This allows different sets of function keys to be mapped for different user groups or for different logical groupings of forms. The labels (shown under the function key name) will be shown underneath the application form providing the “Show function keys” attribute is checked in the **Application Parameters** screen.

## Error Handling

In addition to the log files, system errors are presented to the users using a form which shows relevant information can be used to directly email support. This includes whatever instructions were specified in the **Application parameters** admin screen. The user is presented with options to retry the form, restart the application, log off, send an email to support or view the full stack trace.

This sample error form was produced by shutting down the database during a commit operation.

**IRD**

Status page



The system has encountered an error.

Please [click here](#) to re-try the current form.

[Click here](#) to restart the application.

[Click here](#) to log out.

|                 |                         |
|-----------------|-------------------------|
| Time and Date   | 23:45 November 18, 2006 |
| Session Created | 16:25 November 17, 2006 |
| Station Name    | STN_192_168_1_1         |
| Station IP      | 192.168.1.1             |
| Active Form     | xmlforms/GEM.xml        |
| Username        | anonymous               |

Summary information: null; nested exception is:  
org.jboss.tm.JBossRollbackException: Unable to commit,  
tx=TransactionImpl:XidImpl[FormatId=257, GlobalId=amber.mss/36, BranchQual=  
localId=36] status=STATUS\_NO\_TRANSACTION; - nested throwable:  
(org.jboss.resource.connectionmanager.JBossLocalXAException: could not commit  
local tx; - nested throwable: (org.jboss.resource.JBossResourceException:  
SQLException; - nested throwable: (java.sql.SQLException: Io exception: Broken  
pipe))); - nested throwable: (org.jboss.tm.JBossRollbackException: Unable to  
commit, tx=TransactionImpl:XidImpl[FormatId=257, GlobalId=amber.mss/36,  
BranchQual=  
localId=36] status=STATUS\_NO\_TRANSACTION; - nested  
throwable: (org.jboss.resource.connectionmanager.JBossLocalXAException: could  
not commit local tx; - nested throwable:  
(org.jboss.resource.JBossResourceException: SQLException; - nested throwable:  
(java.sql.SQLException: Io exception: Broken pipe))))))

Please contact Support

You can also [send e-mail](#) to get support.

[Click here](#) to see a stack trace.

## webManager

Webmanager conforms to the standard Model View Controller (MVC) architecture. The webManager J2EE front-end can run in the same J2EE container as the EJB's or can be configured to run on a separate server. Communication between the two is via RPC for performance, passing a single parameter, again, defined by the XML form.

The EJBs correspond to the original online programs and may deal with several transaction types. The EJBs, themselves, exist as stateless session beans which are managed by the container. Several threads corresponding to one program may exist at any one time if the traffic demands it. The threads are available for multiple transactions and are only discarded after a period of inactivity in order to reduce the overhead of start-up, primarily establishing database connections.

## Compatibility

All development has been done using the SunW J2EE-RI Reference container, the reference implementation for all J2EE technology. This means that any mainstream 3rd party J2EE product such as IBM Websphere, Oracle Application Server, JBoss or Tomcat may be used since they have to conform to at least this standard. The migrated code will therefore run out of the box on any J2EE platform. The only restriction is that you use Java 1.5 (the newest version) or later. This is because we have utilised the new "Generics" feature of Java. This shouldn't pose any difficulty since every major vendor has already moved to this version. If authenticating against Microsoft Active Directory, the minimum Java version must be 1.5.08.

No commercial or open-source frameworks are used in the code. This means that the solution is vendor-independent and insulated from future changes in this volatile market.

## Deployment

Ant scripts are supplied for compiling the application and creating an EAR file which is compatible with JBoss and Websphere, the standard deployment targets. All database connections are via container managed data-sources. The resultant EAR file can be directly deployed to the application server. Small modifications to the ant scripts may be required to deploy to other application servers, such as Weblogic.

## Onward Development/Refactoring of the Application

The application is left in a form that is very standard so that full advantage of the latest development tools can be taken. The development tools are outstanding in their functionality and ease of use. It is possible (and easy) to single-step and set breakpoints throughout an entire transaction - even moving seamlessly from the web front-end to the online program, viewing and altering data as you go.

Each form is defined in a single XML file. This form is processed by an XSLT file and displayed on a standard browser with a CSS style sheet. XSLT is a processor (standardised by W3C) which uses the XSL definition file to transform XML to anything else - in this case, to HTML. What this means is that multiple XSL files could be prepared, each designed to produce transformations to different media, such as wireless or print, and the same programs will then be available without any re-coding. The CSS file can be modified to alter fonts, colours and backgrounds while the XSL file can be modified to introduce new features (such as drop down boxes). Different systems (e.g. DEV, TEST and LIVE) can be given different colour schemes etc. The possibilities are endless.

Integrated development tools can be used to good effect to manage future development. Oracle JDeveloper and Eclipse are both used in-house on the development of Jbol itself and can be recommended. These tools provide many useful features for assisting with code refactoring and development.

The overall design of the migrated code lends itself to incremental development. A single global logic can be altered, or changed to use a better Java style code – for example, passing parameters rather than depending on global logic – without affecting other code.

New EJBs can be developed and tested, then when complete, can be incorporated into the main system by merely changing the “bean” name in the XML descriptor file.

## Reliability/Scalability

Another huge benefit of the technology is reliability and scalability.

An online program can crash without affecting any other users and an individual (failing) machine can be removed from the network without shutting down the system. If configured correctly users would not even notice a server crash.

The options for scaling up are no longer limited to increasing the size of the primary server. The container managed applications can be moved to different hardware, possibly running different operating systems, without any change or re-compilation. It is possible to have the online programs simultaneously running on large Unix servers, Linux PC and Windows boxes, all talking to a database which could reside on any number of machines (also with different operating systems) and multiple web-servers. Increasing capacity is merely a matter of adding a new machine to the network, configuring the J2EE container and loading your EAR file. It is not even necessary to bring the system down to accomplish this.