From Vplus to Web Application

CIO Technologies, Inc. is a small software house, specialized in design and implementation of custom mission critical applications. We have created software for travel industry, retail back-office operations, credit card processing, fulfillment and warehouse management. We started using Vplus, Cobol and Image as primary development tools, giving us a very productive environment combined with the rock solid HP3000 hardware. We have also implemented solutions using other languages and technologies, such as C, Pascal, 4GL languages and Lisp. We have found many tools and technologies which are more exciting to work with than Vplus, Cobol and Image but could not find a better toolset for business applications for the HP3000 environment. Our requirements for a good application environment includes

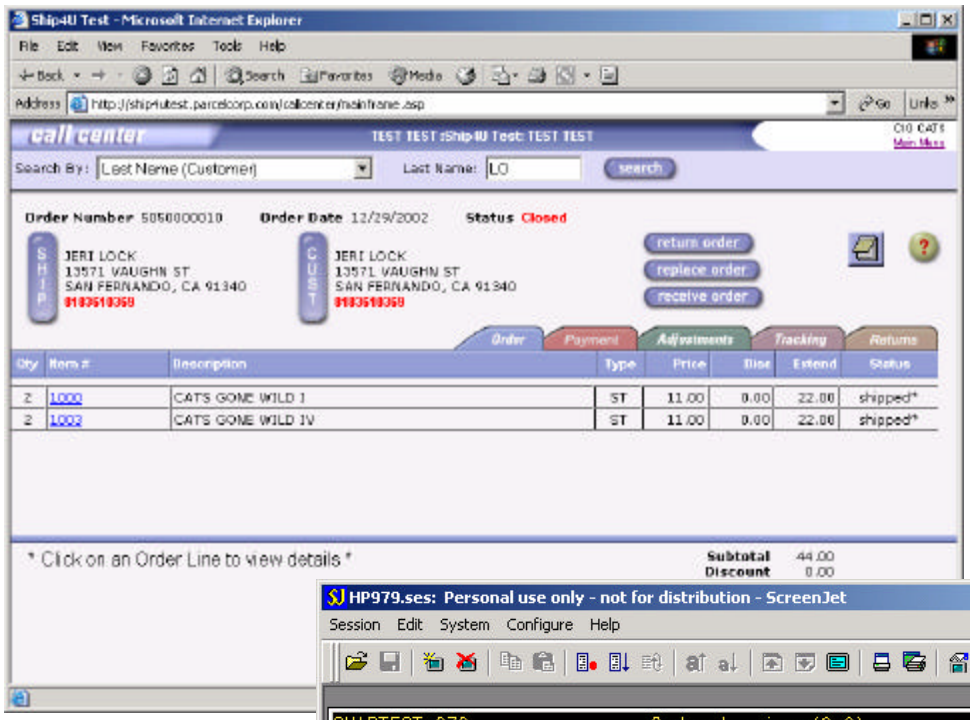
- Rock solid hardware. You should be able to run mission critical application on a single system or use dual systems for 7 x 24 operation.
- A developer should be able to create data bases, write programs and implement production systems without assistance from a large staff of experts.
- No maintenance time should be used required to stay compatible with the last operating system or tool release.
- The daily operation and management of the system and software should be minimal.
- The environment should be easy to interface with other environments
- The system should be protected against misuse, hackers, viruses, etc.
- A good application environment should consist of proven technology.

Our choice of technology made it possible for us to develop and maintain large applications even though our company is very small. The only critical issue we faced was the common IT perception that the HP3000 was a dinosaur of a system and the fact that our Vplus terminal interface strengthened that impression. One of our customers installed a new financial system from a leading vendor; they spent over a year to customize it to handle a complicated internal billing system. Finally they gave up, asked us to move the data to our system (we had them up running in less than a month). Since then the customer has changed IT leadership twice, both times the new CIO started by evaluating whether to replace our old system with the modern system from the application vendor!

We listened to our customers and it was obvious that it was time to find a replacement for Vplus. One of our customers using our Warehouse management and Fulfillment application finally helped fund development of a new user interface. This was the start of a series of project which resulted in not only the new user interface but also an application middleware, Omnihost, and a process to transfer Vplus applications into web applications.

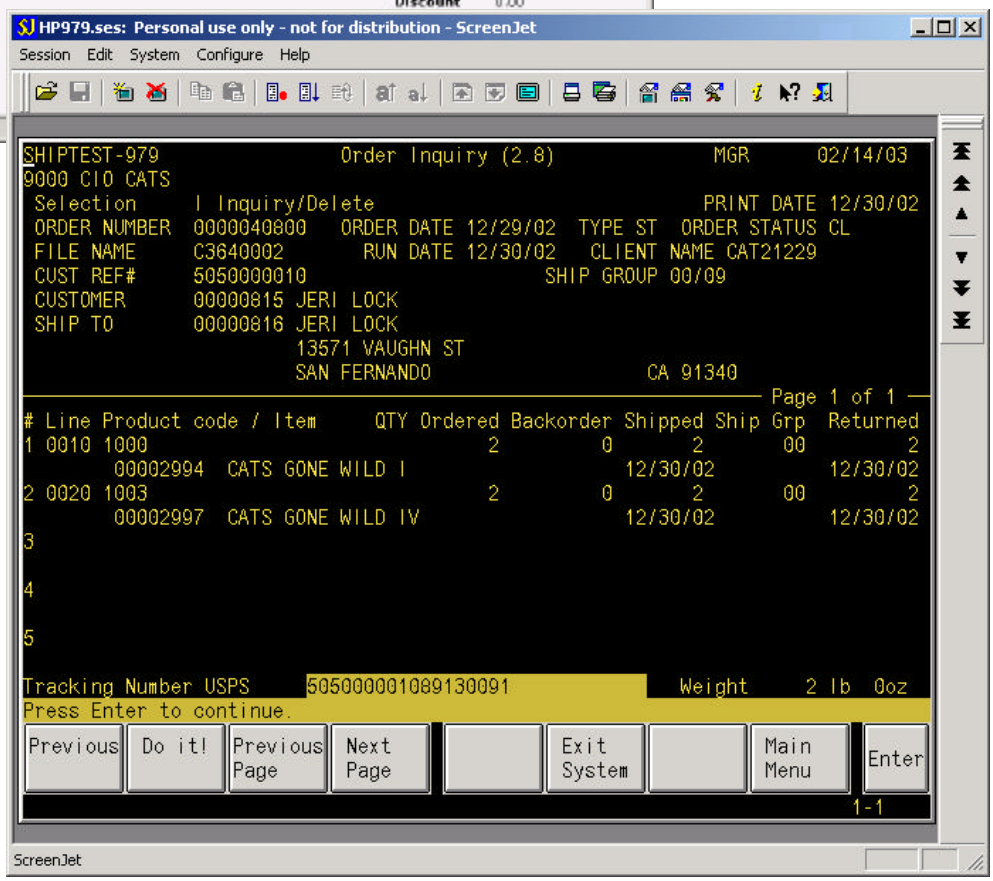
This presentation will discuss experiences from the projects and the frame work. The experiences are general to their nature and useful for other preparing for a migration or

reengineering effort. The examples using Omnihost shows a process to create web application maintaining existing application logic. The work required to transfer the Vplus application is more extensive than a migration but less than a rewrite (due to the fact we maintain the application logic).



This is the new look

This is what our system used to look like.

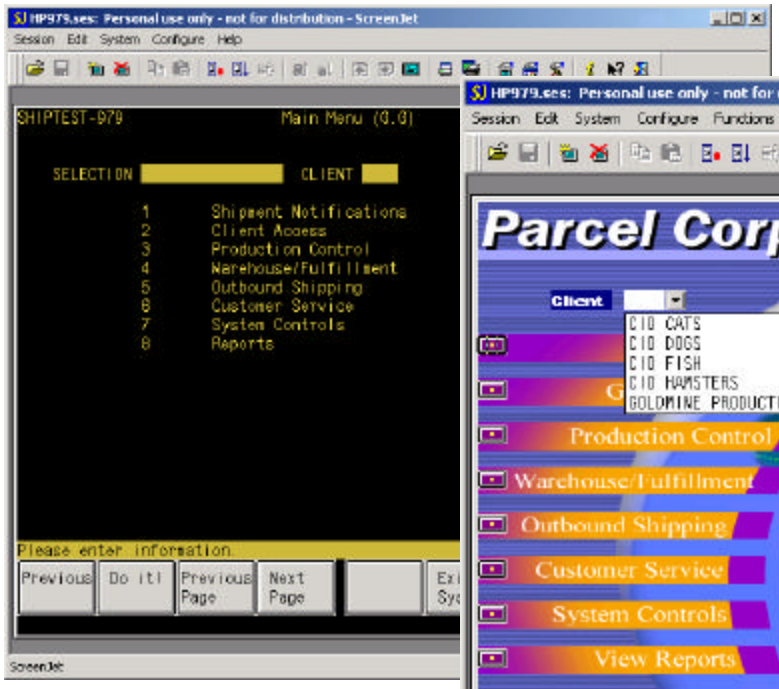


Vplus to Windows Look and Feel

We were not sure what our new user interface should look like, should we go for a Windows or Web look and feel? We looked at different tools and found a very interesting alternative, ScreenJet 3000, from ScreenJet Limited. The tool intercepts Vplus calls and passes the control to a GUI screen defined by a Designer tool. ScreenJet gave us the opportunity to transfer the Vplus screen to a Windows look and feel, using 3D panels, combo boxes, radio buttons, check boxes and macros. It was a major improvement for the users; it took our application as far as we could without changing the 24 x 80 paradigm.

- Cryptic codes were replaced by combo boxes or radio buttons
- Yes/No questions were turned into check boxes
- the function keys replaced by buttons or icons
- macros were used to take Vplus data and link to the web, e.g. Fedex and UPS tracking).

All these changes could be done without changing our COBOL code. We have used ScreenJet 3000 to transfer 225 Vplus screens to GUI format.



Before

After

The new user interface was popular with the users and we planned to keep if for a long time. Hewlett-Packard changed our plans overnight by deciding to discontinue the HP3000 product line. We were now faced with the challenge to move the entire application including the new user interface. It was natural to use our recent experience and start by creating a web application to replace the GUI interface and then as a second step move COBOL code and data base. We knew that our users would not mind running the HP3000 for a few more years as long as the user interface was state of the art. At this time we wanted to get away from the Vplus limitations which could not easily be corrected by ScreeJet, such as;

- the 24 x 80 paradigm – all programs were written with the understanding that the data should be presented in a screen 24 rows long and eighty characters wide. Often similar functions were split into several programs due to space limitations without thinking about user friendliness (e.g. programs for order inquiry, receive returns and order confirmation have almost the same look)
- a user looking at an order wants to be able to see detail information, customer, ship-to, product inventory and other information without leaving the order window
- Large Vplus system requires hierarchical menu system to navigate which is hard for new users and makes it difficult to jump between tasks
- Web users are used to navigate using hyperlinks without going through menus, however hyperlinks are almost impossible to implement in a 24 x 80 character screen
- The web paradigm lets users see all information they can find, the Vplus systems often restricted the users to certain menus
- Secure encrypted Vplus sessions over Internet required us to set up VPN (Virtual Private Network) networks. Unfortunately the VPN technology is not entirely standard so we needed different client software depending on the firewall / VPN installation at the server. SSL (Secure Socket Layer) communication over Internet is much easier to implement than VPN.

One ScreenJet limitation turned out to have been very good for us, the feature set consisting of windows objects was so small that we could handle the design our selves. This is not the case with web design and we discovered early that we needed outside development and design help.

We decided to try to keep the benefits of the HP3000 environment (reliability, performance, simplicity, etc.) but at the same time get rid of all HP3000 traditions which made no sense in the new environment.

Little did we know that our decision made us pursue a very narrow approach, there are numerous tools and frameworks for web applications if are ready to throw away your HP3000 background as well as several tools to migrate the HP3000 paradigm to a new platforms.

The Challenge

- To develop a state of the art web application
- To transfer the rock solid HP3000 architecture to a state-less Internet environment
- To create a cost efficient development environment
- To migrate the application without changing the presentation
- To create a cost effective process to "webify" other Vplus applications

To develop a state of the art web application

Our first step was to try to find out what a "state of the art" web application should look like. To make things worse, almost all the leading Internet companies were falling apart at that time. Obviously the".com" upstarts did not have sound business plans, but they still dominated the web development discussion. The exercise was necessary as the current look and feel was so obvious for us after all years of HP3000 development; it was hard to not think in terms of 24 x 80 screens, instant response and hierarchical menu systems. We were leaving a very successful paradigm for business applications in search of a new.

We used to think about our applications as one entity even though we had learned from the client server discussions that the system should be divided into tiers. We now decided to follow the familiar model with three tiers, presentation, application and data structure. However the analysis revealed that it is more to it than our earlier client server discussion revealed. Not only do we have three tiers but they are different to their nature. In our example warehouse management and fulfillment, the differences are evident

- the data structure has remained fairly constant over a long period of years – a customer order had payment terms, bill-to, ship-to and order lines with product codes, quantity, prices and discounts even before the first commercial computers were built.
- the application logic has a life cycle of typically seven years or more. A warehouse operation does not change drastically in a short time period as it is very expensive to replace shelving, conveyer belts, scales, bar code scanners, etc.
- the presentation layer has a very short life span, many application packages comes out with at least a new version every year to stay competitive and compatible with current trends (client server, windows, web)

This knowledge convinced us that it is both possible and intelligent to replace the presentation layer without rewriting the entire system.

A balanced system should consist of three independent tiers, data structure, application and presentation. The three tiers have different life cycles and should be developed with different tools using standard interfaces between the different parts.

Independent – you should be able to keep the data and replace the application logic, keep the logic and replace the presentation or in case of our final phase in the migration plan – keep presentation and application logic and move the data to a new data base.

Different tools – this is a consequence of the radically different life cycles. You cannot have a state of the art presentation if you don't use the current set of web tools. On the other hand there are no web or windows tools which you can expect to use unchanged five to ten years from now.

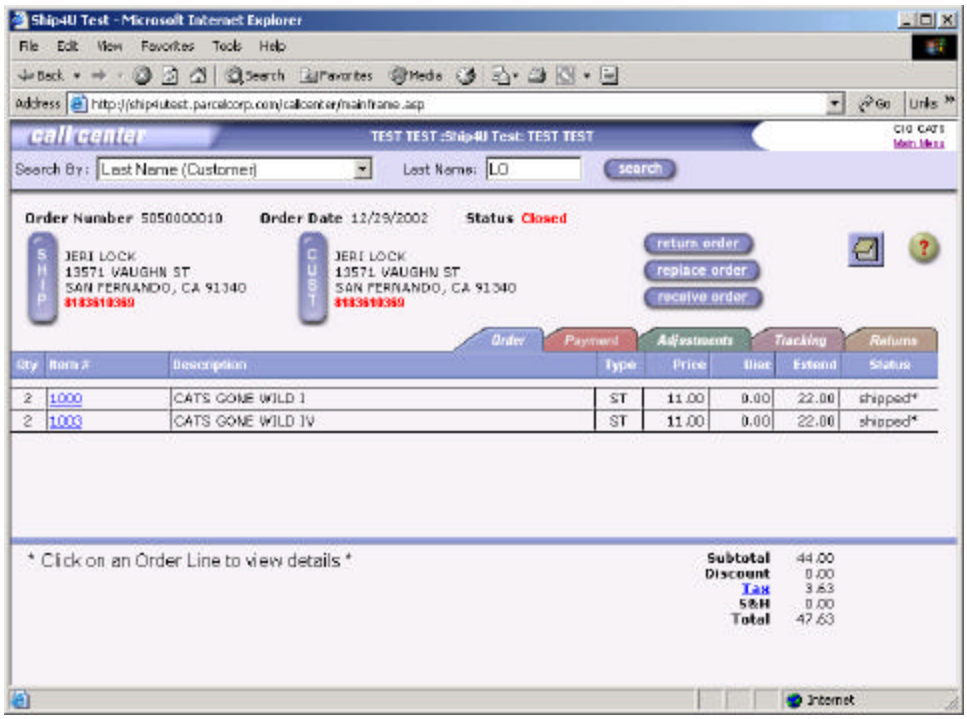
Standard Interfaces – use a standard interface to communicate between the different tiers. It makes it is easy to change one tier without affecting the other and to be able test different tiers independently. For the SHIP4U system we decided to use standard SQL as interface between data structure and application and XML between application and presentation.

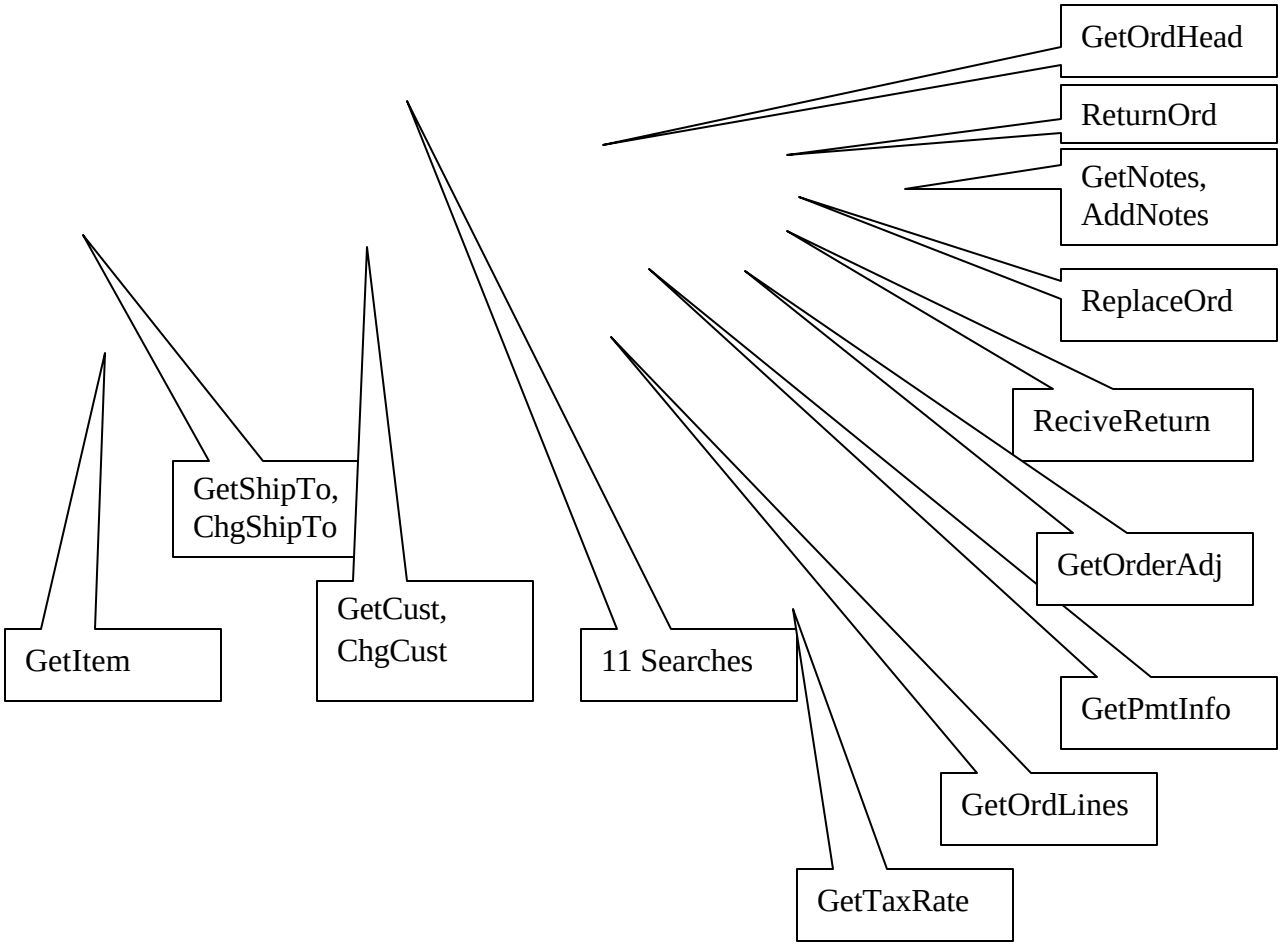
To create a new user interface we need to remove the one to one relationship between programs and Vplus forms (or forms families). It is not acceptable in a web application to look at a customer order and then have to traverse a menu system to be able to return or replace a product in the same order. The user expects to have additional information available instantly, through pop-up screens or new windows. It was the end to our old menu based program structure. How could we keep the application logic? The answer was to isolate the application logic from navigation and Vplus handling. Previously we had one program for customer maintenance; it was replaced by four new programs AddCustomer, ChgCustomer, GetCustomer and DelCustomer. We named the new programs transactions.

The one to one relationship between application program and Vplus form is replaced by a many to many relationship. One transaction can be used in many web screens and one screen can use many transactions.

The decision had a profound impact on the project, we got a clear model to work with and we reduced the size of the HP3000 side project to a fraction of what we initially thought it was going to be. First we could ignore all logic required by Vplus, menu handling and access control (tied to menus). The next step was to eliminate unnecessary redundancy in existing code, order inquiry, confirmation, return handling, etc. all start with an order search, followed access to customer, ship-to and order information this could now be replaced with search transactions, GetCustomer, GetShipTo, GetOrderHeader and GetOrderLines. We estimate that the core application logic we needed to preserve amounts to 10 – 15% of our original COBOL code.

The typical Vplus hierarchical menu system is flattened out, everyone who has access to order inquiry can see the same data and the former separate programs have been replaced with buttons, tabs and pop-up windows.


Ship4U Test - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://ship4utest.parcelcorp.com/callcenter/mainframe.asp
call center
TEST TEST :Ship4U Test: TEST TEST
Search By: Last Name (Customer)
Last Name: LO
search
Order Number 5050000010
Order Date 12/29/2002
Status Closed
SHIP
JERI LOCK
13571 VAUGHN ST
SAN FERNANDO, CA 91340
CUST
JERI LOCK
13571 VAUGHN ST
SAN FERNANDO, CA 91340
return order
replace order
receive order
Order
Payment
Adjustments
Tracking
Returns
Qty Item # Description Type Price Disc Extend Status
2 1000 CATS GONE WILD I ST 11.00 0.00 22.00 shipped*
2 1003 CATS GONE WILD IV ST 11.00 0.00 22.00 shipped*
* Click on an Order Line to view details *
Subtotal 44.00
Discount 0.00
Tax 3.63
S&H 0.00
Total 47.63
Internet



The Web Order screen shows 16 of the transactions used in the view. In a Vplus based system that would correspond to 10+ programs. This also changes the way the access control is done as the restriction who can return an order can not be linked to a menu, instead it has to be linked to a pushbutton in this case.

To transfer the rock solid HP3000 architecture to a state-less Internet environment

The old Vplus system was designed for terminals connected directly to the HP3000 which created a very robust environment. When we started using Windows and terminal emulators we encountered a new issue; the user session could instantly be interrupted by closing the window or by a Windows itself. The web adds many new dimensions, the data is transmitted over public networks and must be encrypted, there can be delays caused by other activities (someone is using all Internet bandwidth by downloading a movie), and there are many layers of network equipments and servers you are dependent on without having any chance to control. On top of that the typical web mode of operation is state-less while the old applications are context sensitive (you log-on once, the customer maintenance program displays the old record before you can change it, etc).

The only things guaranteed in an Internet connection are that you will lose information and that there is a potential for sabotage.

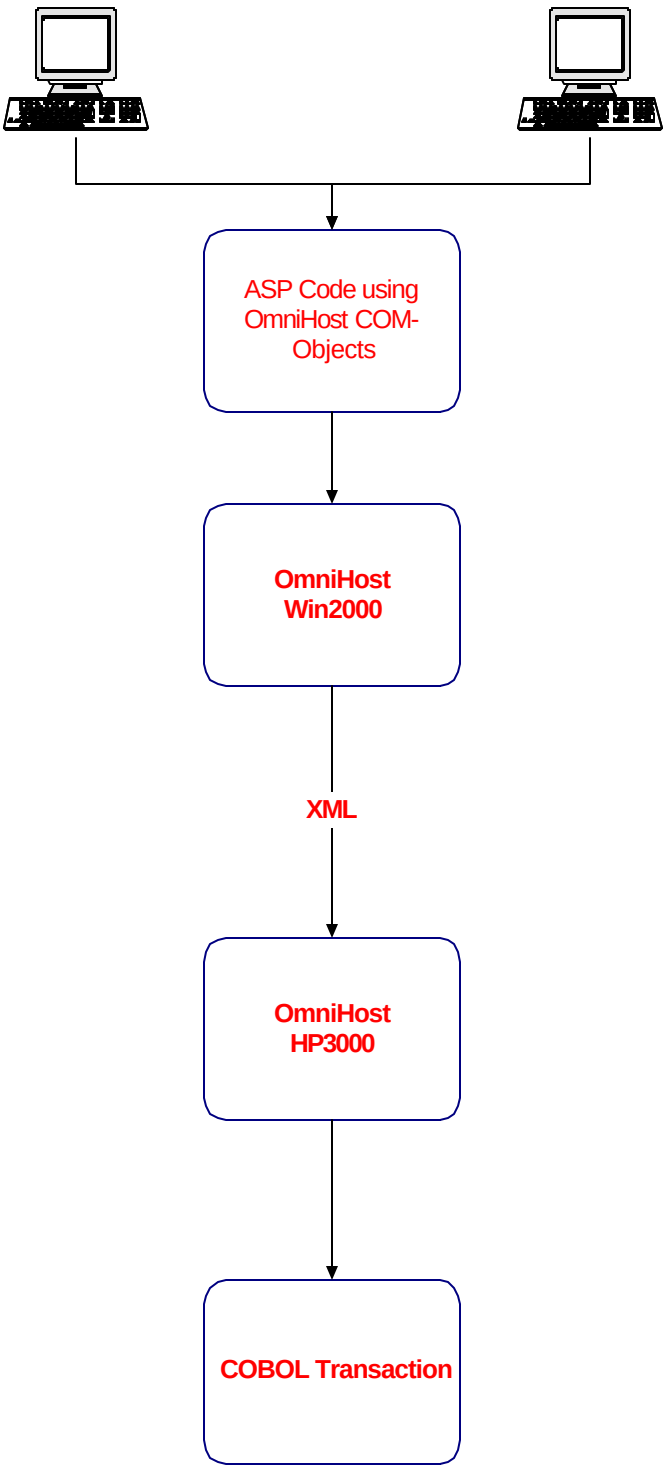
We realized that we could not make Internet completely fail-safe so we changed our ambition to - how can we protect data integrity in our application using an unreliable connection? The answer was again a paradigm shift; our transactions needed to be stateless. The transactions were already restricted to one task, now we had to go a step further to make them binary – either the task was performed correctly or it was not performed at all. In addition we needed a new mechanism for the user to see if an update took place or not.

The Internet is open for sabotage, so the application needs to be much more secure than our traditional Vplus code. This is achieved by several layers; first the system is using encryption, SSL, secondarily the web-server has only presentation logic and no data. The Data resides on the application (or data base) server, which is not directly accessible from web. We don't even allow ODBC access from the web server as it could give a hacker an easy way to get the data. The final layer, authentication and access control is performed entirely on the application server.

To create a cost efficient development environment

The challenge is to be able to keep a small project group; to allow the web designers to create the presentation layer without knowing the HP3000 and let the COBOL programmers concentrate on the application logic.

A successful application is a balance between technique, presentation and business logic. Keep it as simple as possible, but not simpler.



Application development has a difficult paradox; the best systems are so natural to the users that they hardly notice the design behind it. It is easy to create complex hard to use and maintain systems but very difficult to create simple systems without sacrificing functionality. One very important aspect of our Vplus, COBOL and Image environment was its technological simplicity; everyone could concentrate on the application instead of the technique. In the new environment the application focus is threatened by a constantly changing technology.

The web environment offers many design tools and the portfolio changes frequently. We learned by mistakes that we needed standards for web design to restrict our web developers. It can be good to use a lot of features to create attention to your web site but it is not cost effective to replace 225 Vplus screens without standards and it would be very confusing for the users.

At the same time we did not want our COBOL developers to think about how the customer or order would be presented on the web so we had to implement new standards for our COBOL application.

Our conclusion was that we had to isolate both our COBOL and web developers from most of the technology by hiding it in a middleware and the standard based interface between presentation and application.

To migrate the application without changing the presentation

This issue worried us at the start, we do not want to change the presentation again when migrating the application and data base to another platform. However, our previous discussion leading to state-less transactions, middleware and strict interfaces solved that issue. The ASP code on the web server does not even know where the data comes from, so once the HP3000 middleware (which is very compact) is moved to the new platform the transactions could come from the new platform instead.

```
 12:44:03 2I WebMain -2  769 GetUser      N
: Buf len +000000318

:  [<?xml version="1.0" encoding="UTF-8"?>
                                          <SHIP4U><UDPBuf Ip-re]
:  [ply="" Socket-reply="" Request="GetUser" Request-date-time="]
:  [2003-04-15T12:45:32" Transaction-no="100308" Sequence-no="1"]
:  [ Client-id="" Call-center="PCA" User-id="GUNNARF" Max-lines=]
:  ["25"></UDPBuf><GetUser User-id="GUNNARF" Call-center="PCA"><]
:  [/GetUser></SHIP4U>]

12:44:03 3I WebMain -2  769 GetUser      N    1 1
: Buf len +000000857

:  [<?xml version="1.0"?><SHIP4U><UDPBuf Request="GetUser" Reply]
:  [-date-time="2003-04-15T12:44:03" Transaction-no="100308" Seq]
:  [uence-no="1" Status="ok" Sequence-status="Unique" Client-id=]
:  ["" Call-center="PCA" User-id="GUNNARF" Program="WebMain -2" ]
:  [Sub-program="CS-GETUSER" /><GetUser User-id="GUNNARF" Call-c]
:  [enter="PCA" Name="Gunnar Fredlund" Title="CEO" Telephone="80]
:  [5-898-2444" Ext="" Home-telephone="805-682-5572" E-mail="gun]
:  [narf.ciotech.com" Start-date="1987-08-27" Finish-date="" Rea]
:  [son="" Printer="LP" Location="0001" Client-id="" Account-mgr]
:  [-sw="Y" Call-center-access="Y" Client-set-up-access="M" Syst]
:  [em-set-up-access="M" Reports-access="M" Cancel-ord-access="M]
:  [" Change-ord-access="M" Refund-access="M" Return-access="M" ]
:  [Exchange-access="M" Replacement-access="M" Create-ord-access]
:  [="M" Continuity-access="M" System-name="TEST4U" Serial-acces]
:  [s="M" /></SHIP4U>]
```

The example shows the XML communication between the HP3000 and the web-server to get user information. The web server can receive the exact same XML format from a Linux or Win2000 server, and cannot tell any differences..

The concept is so powerful that the transactions does not need to come from one platform but can be mixed HP3000, Linux and Win2000. This allow us to migrate one module at a time and also give us a new way to implement batch jobs, the batch job can create the XML and call the transaction as well as the web browser.

The code sample shows an extract from a batch program using the ReciveReturn transaction to return all customer orders listed in a batch file. It shows how easy it is to create the XML format and to uses cudpsend to send the XML-transaction and sudpread to receive the reply from the application.

```
     MOVE "All-lines-sw"            TO XML-OUT-ATR-LABEL (3).
     MOVE "Y"                       TO XML-OUT-ATR-VALUE (3).

     MOVE "Line-no"                 TO XML-OUT-ATR-LABEL (4).
     MOVE " "                       TO XML-OUT-ATR-VALUE (4).

     MOVE "Return-qty"              TO XML-OUT-ATR-LABEL (5).
     MOVE " "                       TO XML-OUT-ATR-VALUE (5).

     MOVE "Return-restock"          TO XML-OUT-ATR-LABEL (6).
     MOVE "N"                       TO XML-OUT-ATR-VALUE (6).

     MOVE "Return-goods"            TO XML-OUT-ATR-LABEL (7).
     MOVE "N"                       TO XML-OUT-ATR-VALUE (7).

     MOVE "Received-loc"            TO XML-OUT-ATR-LABEL (8).
     MOVE " "                       TO XML-OUT-ATR-VALUE (8).

     MOVE "Authorization"           TO XML-OUT-ATR-LABEL (9).
     MOVE "WEBMGR"                  TO XML-OUT-ATR-VALUE (9).

     MOVE "Mailing-date"            TO XML-OUT-ATR-LABEL (10).
     MOVE DA-CURRENT-DATE           TO WS-DATE.
     MOVE " "                       TO XML-OUT-ATR-VALUE (10).
     STRING WS-YEAR "-" WS-MONTH "-" WS-DAY DELIMITED BY SIZE
                                   INTO XML-OUT-ATR-VALUE (10).

     MOVE "RMA-no"                  TO XML-OUT-ATR-LABEL (11).
     MOVE " "                       TO XML-OUT-ATR-VALUE (11).

     MOVE 11                        TO XML-OUT-AT-NO.
     CALL "X-OUT-ADD-ELEMENT" USING GLOB XML-OUT XML-BUF.

*** FINISH XML DOCUMENT
     PERFORM                        CAB000-FINISH-XML.

*    CALL "X-PRINT-BUF" USING GLOB XML-BUF XML-OUT-LEN

*** SEND REQUEST ***
     CALL "cudpsend" USING UDP-SOCKET-OUT UDP-PORT-OUT
        PARM-IP XML-BUF XML-OUT-LEN GIVING UDP-STAT
     IF UDP-STAT <> 0
        DISPLAY "cudpsend error " UDP-STAT
        STOP RUN
     END-IF.

*** RECEIVE REPLY ***
     MOVE UDP-MAX-LEN               TO XML-OUT-LEN
     COMPUTE UDP-CLEN = FUNCTION LENGTH (UDP-CLIENT-ADDR)
     CALL "sudpread" USING UDP-SOCKET-IN UDP-CLIENT-ADDR
        UDP-CLEN UDP-TIMEOUT XML-BUF XML-OUT-LEN
        GIVING UDP-STAT.
     IF UDP-STAT <> 0
        DISPLAY "cudpread error " UDP-STAT
        CALL "FATAL-ERROR" USING GL-ERR-MSG
     END-IF.
```

To create a cost effective process to "webify" other Vplus applications

This step is an ongoing process as we have many Vplus screens to transfer and due to interest from other installations. The framework is also used extensively to add new features and modules to the system. Our first thought was to create an automatic translation between our existing COBOL code and the new web transactions. We more or less abandoned that approach when we realized how tiny the application portion was in our Vplus programs. In our old COBOL programs we had used separate sections for data validation which made it easy to isolate the application logic and copy it to the new transaction structure. Instead we have been working to streamline our middleware, COBOL and Web standards to implement features such as access control in middleware, sub programs for application logging, and debugging and monitoring.

The process to create a web application from a Vplus system using the OmniHost middleware can be summarized as follows

1. Define which transactions you need in the web system and which current COBOL program has a corresponding functionality and isolate the application logic from existing COBOL programs
2. Chose an existing transaction as starting point and add the code specific for the transaction
3. Add a test-template in "WebCons" to test the transactions on your host system before the web-development is ready
4. Add a call to the transaction (COBOL sub-program) in "WebMain" and link WebMain
5. Write an ASP script using the transaction
6. Test functionality using "WebMon"

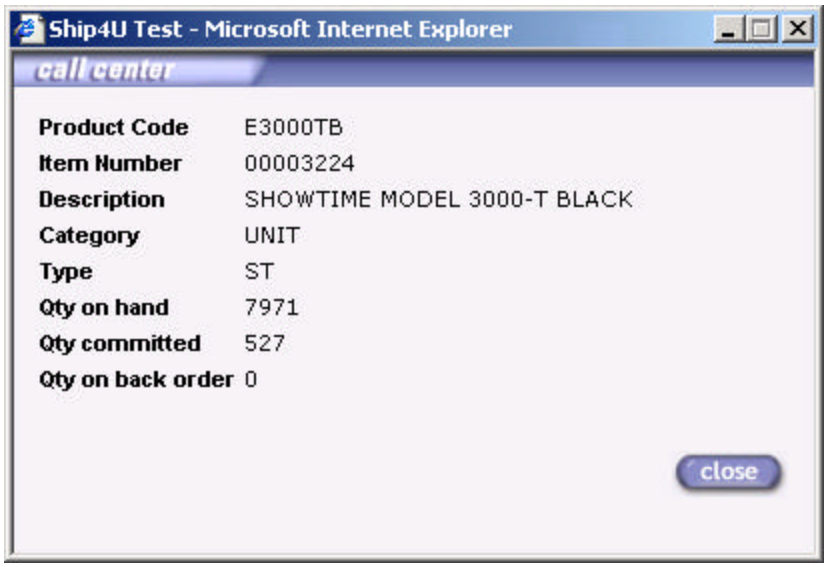
The critical factor so far has been the web development; the COBOL coding has been fast and easy. It has been a major challenge to find a good standard for web applications, and it has just recently been implemented. With the new standard in place the web development should be much more efficient and it will be time to look at automating the COBOL process.

Summary

It is possible to transfer a Vplus system into a modern looking web-application in a cost effective manner and at the same time minimize the risks. The process maintains the old application logic, however the result will appear as a rewrite as only the core logic is kept from the original system. The process involves much more work than a strict migration but the user interface and functionality will be superior. When comparing to a system rewrite process this approach is much easier, the old system is reduced to 100,000 lines of code instead of 1,000,000, and on top of that the risks are minimized by applying a proven step by step process instead of being faced with hundreds of technical decisions.

Appendix – Item-Pop-Up

Layout



ASP-Code

```
<% Response.Expires = 0 %>
<HTML>
<HEAD>
<TITLE><%= Application("strAppName") %></TITLE>
<LINK href="style.css" rel="StyleSheet" type="text/css">
</HEAD>
<BODY bgColor="#F3F1F6" leftMargin="0" rightMargin="0" bottomMargin="0"
topMargin="0">
<%
Dim hst
Dim ItemNo

ItemNo = Request.QueryString("ItemNo")
Set hst = Server.CreateObject("OmniHost3.OHClient")
hst.Profile = Application("dbProfile")
hst.UserID = Session("UserID")
hst.ClientID = Session("ClientID")
hst.Request = "GetItem"
hst.SetParam "Product-code", ItemNo
hst.Execute
'Response.Write Replace(hst.XMLResponse, "<" , "&lt;")
If hst.Error Then
      Response.Write hst.LastError
Else
%>
<TABLE border=0 cellSpacing=0 cellPadding=0 width="100%">
<TR><TD width="141"><IMG src="images/header_title_small.gif" width="141"
height="17" alt="" border="0"></TD>
      <TD width="*"><IMG src="images/header_bar_small.gif" width="100%"
height="17" alt="" border="0"></TD></TR>
</TABLE>
<TABLE border=0 cellSpacing=0 cellPadding=10 width="100%">
```

```
<TR><TD>
<TABLE>
<TR><TD class="hd">Product Code</TD><TD><%= hst.Field("Product-code")
%></TD></TR>
<TR><TD class="hd">Item Number</TD><TD><%= hst.Field("Item-no")
%></TD></TR>
<TR><TD class="hd" vAlign=TOP>Description</TD><TD><%=
hst.Field("Description") %><BR><%= hst.Field("Description2") %></TD></TR>
<TR><TD class="hd">Category</TD><TD><%= hst.Field("Product-cat")
%></TD></TR>
<TR><TD class="hd">Type</TD><TD><%= hst.Field("Item-type") %></TD></TR>
<TR><TD class="hd">Qty on hand</TD><TD><%= hst.Field("Qty-on-hand")
%></TD></TR>
<TR><TD class="hd">Qty committed</TD><TD><%= hst.Field("Qty-committed")
%></TD></TR>
<TR><TD class="hd">Qty on back order</TD><TD><%= hst.Field("Qty-on-
backorder") %></TD></TR>
</TABLE>
<BR><BR>
<IMG align=RIGHT src="images/close_up.jpg" width="58" height="23" alt=""
border="0" onMouseOver="this.src='images/close_over.jpg'"
onMouseOut="this.src='images/close_up.jpg'" onClick="window.close()">
</TD></TR>
</TABLE>
<%
End If
hst.Close
Set hst = Nothing
%>
</BODY>
</HTML>
```

XML- communication

```
14:17:06 2I WebMain -2  874 GetItem      N
      : Buf len +000000312

      :  [<?xml version="1.0" encoding="UTF-8"?>
                                  <SHIP4U><UDPBuf Ip-re]
      :  [ply="" Socket-reply="" Request="GetItem" Request-date-time="]
      :  [2003-04-15T14:18:36" Transaction-no="100412" Sequence-no="1"]
      :  [ Client-id="0008" Call-center="PCA" User-id="GUNNARF" Max-li]
      :  [nes="25"></UDPBuf><GetItem Product-code="EST03B5PAY"></GetIt]
      :  [em></SHIP4U>]

14:17:07 3I WebMain -2  874 GetItem      N    1 1
      : Buf len +000001378

      :  [<?xml version="1.0"?><SHIP4U><UDPBuf Request="GetItem" Reply]
      :  [-date-time="2003-04-15T14:17:07" Transaction-no="100412" Seq]
      :  [uence-no="1" Status="ok" Sequence-status="Unique" Client-id=]
      :  ["0008" Call-center="PCA" User-id="GUNNARF" Program="WebMain ]
      :  [-2" Sub-program="CS-GETITEM" /><GetItem Item-no="00003224" P]
      :  [roduct-code="E3000TB" Description="SHOWTIME MODEL 3000-T BLA]
      :  [CK" Description2="" Product-cat="UNIT" Item-type="ST" Item-s]
      :  [ub-type="" Size="" Color="" Style="" Delivery-conf-sw="Y" Ai]
      :  [r-sw="Y" Printed-material-sw="N" Special-standard-sw="N" Ove]
      :  [rsized-sw="N" Ship-separate-sw="Y" Inc-accessory-sw="Y" Ovr-]
      :  [shipping-sw="N" Ship-group-id="69" Ship-method-code="" Give-]
      :  [away-sw="N" Sell-below-sw="N" Price="0.00" Std-cost="0.00" S]
      :  [hipping-cost="0.00" Qty-on-hand="7971" Qty-committed="527" Q]
      :  [ty-on-backorder="0" Qty-on-open-po="0" Qty-on-hold="0" Seria]
```

```
: [l-no-req-sw="N" Required-level="0" Weight="23.06" Unit-of-me]
: [asure="#" Units-per-box="1" Backorder-sw="Y" Bill-of-materia]
: [l-sw="N" Catalog-item-sw="Y" Start-date="2002-11-08" Dimensi]
: [on-width="15.30" Dimension-length="19.40" Dimension-height="]
: [15.60" Special-handling-sw="N" Taxable-sw="Y" Free-space-wid]
: [th="0.00" Free-space-length="0.00" Free-space-height="0.00" ]
: [Reverse-kit="00003225" Max-weight="0.00" Discontinued-sw="N"]
: [ Max-line-qty="1" Non-machine-sw="N" Insert-prod-1="" Insert]
: [-prod-2="" Max-order-qty="0" Lot-control-sw="" /></SHIP4U>]
```

Cobol-Program

```
*****************************************************************
*     P R O P R I E T A R Y   P R O G R A M   M A T E R I A L   *
*                                                               *
*     This material is proprietary to CIO Technologies, Inc.    *
*     and is not to be reproduced, used or disclosed except     *
*     in accordance with a software license agreement or        *
*     upon written authorization of                             *
*                                                               *
*      CIO Technologies, Inc.                                   *
*      351 S. Hitchcock Way, Suite B-140                        *
*      Santa Barbara, CA 93105                                  *
*                                                               *
*                C O P Y R I G H T   (C)  1991 - 2002.          *
*****************************************************************

*****************************************************************
*                                                               *
* System:  Ship4u                                               *
* Program: GetItem                                 Source: GITEM    *
* Author:  Hans Backman / CIO technologies        Date: 01/03/02 *
*                                                               *
* Customer service request 'GetItem'. Return item information.   *
*                                                               *
* Sign Date     Change                                          *
* CAC  03/26/03 Add remaining item-detail fields                *
*                                                               *
*****************************************************************

$CONTROL DYNAMIC, POST85, BOUNDS, NOWARN
 IDENTIFICATION DIVISION.
 PROGRAM-ID.    CS-GETITEM.

 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01  WS-ELEMENT                      PIC X(32).
 01  WS-ATTRIBUTE                    PIC X(32).
 01  WS-VALUE                        PIC X(250).
 01  WS-TIME                         PIC X(6).

 01  ITEM            COPY            ITEM     OF COPYLIB.
 01  ITEM-ID-XREF    COPY            ITIDXREF OF COPYLIB.

 01  XML-GET-MODE                    PIC X(6).

 LINKAGE SECTION.
 01  GLOB            COPY            GLOBAL OF COPYLIB.
 01  DBAREA          COPY            DBAREA   OF COPYLIB.
 01  XML-IN          COPY            XMLIN  OF COPYLIB.
 01  XML-IN-BUF      COPY            XMLBUF OF COPYLIB.
```

```
 01  XML-OUT         COPY            XMLOUT OF COPYLIB.
 01  XML-OUT-BUF     COPY            XMLBUF OF COPYLIB.
 01  UDP-BUF         COPY            UDPBUF OF COPYLIB.

 PROCEDURE DIVISION USING GLOB DBAREA
    XML-IN XML-IN-BUF XML-OUT XML-OUT-BUF UDP-BUF.
 A000-CONTROL SECTION.

     MOVE "CS-GETITEM"               TO UPB-SUB-PROGRAM.

*** GET ALL ATTRIBUTES FOR ELEMENT 'GetItem' ***
     MOVE "GetItem"                  TO WS-ELEMENT.
     MOVE "FIRST"                    TO XML-GET-MODE.
     CALL "X-IN-GET-ATTRIBUTES" USING GLOB XML-IN XML-IN-BUF
        XML-GET-MODE WS-ELEMENT.
     IF NOT GL-STAT-OK
        CALL "CS-ERROR" USING GLOB XML-OUT XML-OUT-BUF UDP-BUF
        GOBACK
     END-IF.

*** GET VALUE FOR ATTRIBUTE 'Order-no' ***
     MOVE "Product-code"             TO WS-ATTRIBUTE.
     CALL "X-IN-GET-VALUE" USING GLOB XML-IN XML-IN-BUF
        WS-ATTRIBUTE WS-VALUE.
     IF NOT GL-STAT-OK
        CALL "CS-ERROR" USING GLOB XML-OUT XML-OUT-BUF UDP-BUF
        GOBACK
     END-IF.

     MOVE UPB-CLIENT-ID              TO ID-PRODUCT-CODE.
     MOVE WS-VALUE                   TO ID-PRODUCT-CODE (5:30).
     MOVE GL-LOCATION                TO LD-LOCATION
                                        LD-ITEM-LOC(9:4).

*** READ ORDER. RETURN UDPBuf ONLY IF ERROR ELSE RETURN ORDER ***
     MOVE "Unique"                   TO UPB-SEQUENCE-STATUS
     CALL "GET-ITEM-ALIAS" USING GLOB DBAREA ITEM ITEM-ID-XREF.
     IF NOT GL-STAT-OK
        CALL "CS-ERROR" USING GLOB XML-OUT XML-OUT-BUF UDP-BUF
     ELSE
        MOVE "ok"                    TO UPB-STATUS
        CALL "X-OUT-INIT-XML" USING GLOB XML-OUT XML-OUT-BUF
           UDP-BUF
        PERFORM                      B000-EDIT-ITEM
        CALL "X-OUT-FINISH-XML" USING GLOB XML-OUT XML-OUT-BUF
        CALL "CS-WRITE-OUTFILE" USING GLOB XML-OUT UDP-BUF
     END-IF.

     GOBACK.

$PAGE "B000-EDIT-ITEM"
*****************************************************************
* Edit ITEM information.
*****************************************************************
 B000-EDIT-ITEM SECTION.

     MOVE "GetItem"                  TO XML-OUT-ELEMENT.

     MOVE "Item-no"                  TO XML-OUT-ATR-LABEL (1).
     MOVE ID-ITEM-NO                 TO XML-OUT-ATR-VALUE (1).
     MOVE "Product-code"             TO XML-OUT-ATR-LABEL (2).
     MOVE ID-PRODUCT-CODE (5:30)     TO XML-OUT-ATR-VALUE (2).
     MOVE "Description"              TO XML-OUT-ATR-LABEL (3).
```

```
MOVE ID-ITEM-DESC              TO XML-OUT-ATR-VALUE (3).
MOVE "Description2"            TO XML-OUT-ATR-LABEL (4).
MOVE ID-ITEM-DESC2             TO XML-OUT-ATR-VALUE (4).
MOVE "Product-cat"             TO XML-OUT-ATR-LABEL (5).
MOVE ID-PRODUCT-CATEGORY       TO XML-OUT-ATR-VALUE (5).
MOVE "Item-type   "            TO XML-OUT-ATR-LABEL (6).
MOVE ID-ITEM-TYPE              TO XML-OUT-ATR-VALUE (6).
MOVE "Item-sub-type"           TO XML-OUT-ATR-LABEL (7).
MOVE ID-ITEM-SUB-TYPE          TO XML-OUT-ATR-VALUE (7).
MOVE "Size"                    TO XML-OUT-ATR-LABEL (8).
MOVE ID-SIZE                   TO XML-OUT-ATR-VALUE (8).
MOVE "Color"                   TO XML-OUT-ATR-LABEL (9).
MOVE ID-COLOR                  TO XML-OUT-ATR-VALUE (9).

MOVE "Style"                   TO XML-OUT-ATR-LABEL (10).
MOVE ID-STYLE                  TO XML-OUT-ATR-VALUE (10).
MOVE "Delivery-conf-sw"        TO XML-OUT-ATR-LABEL (11).
MOVE ID-DELIVERY-CONF-SW       TO XML-OUT-ATR-VALUE (11).
MOVE "Air-sw"                  TO XML-OUT-ATR-LABEL (12).
MOVE ID-AIR-SW                 TO XML-OUT-ATR-VALUE (12).
MOVE "Printed-material-sw"     TO XML-OUT-ATR-LABEL (13).
MOVE ID-PRINTED-MATER-SW       TO XML-OUT-ATR-VALUE (13).
MOVE "Special-standard-sw"     TO XML-OUT-ATR-LABEL (14).
MOVE ID-SPECIAL-STANDARD       TO XML-OUT-ATR-VALUE (14).
MOVE "Oversized-sw"            TO XML-OUT-ATR-LABEL (15).
MOVE ID-OVERSIZED-SW           TO XML-OUT-ATR-VALUE (15).
MOVE "Ship-separate-sw"        TO XML-OUT-ATR-LABEL (16).
MOVE ID-SHIP-SEPARATE-SW       TO XML-OUT-ATR-VALUE (16).
MOVE "Inc-accessory-sw"        TO XML-OUT-ATR-LABEL (17).
MOVE ID-INC-ACCESSORY-SW       TO XML-OUT-ATR-VALUE (17).
MOVE "Ovr-shipping-sw"         TO XML-OUT-ATR-LABEL (18).
MOVE ID-OVR-SHIPPING-SW        TO XML-OUT-ATR-VALUE (18).
MOVE "Ship-group-id"           TO XML-OUT-ATR-LABEL (19).
MOVE ID-SHIP-GROUP-ID          TO XML-OUT-ATR-VALUE (19).
MOVE "Ship-method-code"        TO XML-OUT-ATR-LABEL (20).
MOVE ID-SHIP-METHOD-CODE       TO XML-OUT-ATR-VALUE (20).

MOVE "Give-away-sw"            TO XML-OUT-ATR-LABEL (21).
MOVE ID-GIVE-AWAY              TO XML-OUT-ATR-VALUE (21).
MOVE "Sell-below-sw"           TO XML-OUT-ATR-LABEL (22).
MOVE ID-SELL-BELOW-SW          TO XML-OUT-ATR-VALUE (22).
MOVE "Price"                   TO XML-OUT-ATR-LABEL (23).
MOVE LD-ITEM-PRICE             TO NA-NUM.
CALL "NUM-TO-TEXT"             USING NUM-AREA.
MOVE NA-TEXT                   TO XML-OUT-ATR-VALUE (23).
MOVE "Std-cost"                TO XML-OUT-ATR-LABEL (24).
MOVE LD-STD-COST               TO NA-NUM.
CALL "NUM-TO-TEXT"             USING NUM-AREA.
MOVE NA-TEXT                   TO XML-OUT-ATR-VALUE (24).
MOVE "Shipping-cost"           TO XML-OUT-ATR-LABEL (25).
MOVE LD-SHIPPING-COST          TO NA-NUM.
CALL "NUM-TO-TEXT"             USING NUM-AREA.
MOVE NA-TEXT                   TO XML-OUT-ATR-VALUE (25).
MOVE "Qty-on-hand"             TO XML-OUT-ATR-LABEL (26).
MOVE LD-QTY-ON-HAND            TO NA-NUM.
MOVE 0                         TO NA-DECIMAL-LEN.
CALL "NUM-TO-TEXT"             USING NUM-AREA.
MOVE NA-TEXT                   TO XML-OUT-ATR-VALUE (26).
MOVE "Qty-committed"           TO XML-OUT-ATR-LABEL (27).
MOVE LD-QTY-COMMITTED          TO NA-NUM.
MOVE 0                         TO NA-DECIMAL-LEN.
CALL "NUM-TO-TEXT"             USING NUM-AREA.
MOVE NA-TEXT                   TO XML-OUT-ATR-VALUE (27).
```

```
MOVE "Qty-on-backorder"         TO XML-OUT-ATR-LABEL (28).
MOVE LD-QTY-ON-BACKORDER        TO NA-NUM.
MOVE 0                          TO NA-DECIMAL-LEN.
CALL "NUM-TO-TEXT"              USING NUM-AREA.
MOVE NA-TEXT                    TO XML-OUT-ATR-VALUE (28).
MOVE "Qty-on-open-po"           TO XML-OUT-ATR-LABEL (29).
MOVE LD-QTY-ON-OPEN-PO          TO NA-NUM.
MOVE 0                          TO NA-DECIMAL-LEN.
CALL "NUM-TO-TEXT"              USING NUM-AREA.
MOVE NA-TEXT                    TO XML-OUT-ATR-VALUE (29).
MOVE "Qty-on-hold"              TO XML-OUT-ATR-LABEL (30).
MOVE LD-QTY-ON-HOLD             TO NA-NUM.
MOVE 0                          TO NA-DECIMAL-LEN.
CALL "NUM-TO-TEXT"              USING NUM-AREA.
MOVE NA-TEXT                    TO XML-OUT-ATR-VALUE (30).

MOVE "Serial-no-req-sw"         TO XML-OUT-ATR-LABEL (31).
MOVE ID-SERIAL-NO-REQ-SW        TO XML-OUT-ATR-VALUE (31).
MOVE "Required-level"           TO XML-OUT-ATR-LABEL (32).
MOVE ID-REQUIRED-LEVEL          TO NA-NUM.
MOVE 0                          TO NA-DECIMAL-LEN.
CALL "NUM-TO-TEXT"              USING NUM-AREA.
MOVE NA-TEXT                    TO XML-OUT-ATR-VALUE (32).
MOVE "Weight"                   TO XML-OUT-ATR-LABEL (33).
MOVE ID-WEIGHT                  TO NA-NUM.
CALL "NUM-TO-TEXT"              USING NUM-AREA.
MOVE NA-TEXT                    TO XML-OUT-ATR-VALUE (33).
MOVE "Unit-of-measure"          TO XML-OUT-ATR-LABEL (34).
MOVE ID-UNIT-OF-MEASURE         TO XML-OUT-ATR-VALUE (34).
MOVE "Units-per-box"            TO XML-OUT-ATR-LABEL (35).
MOVE ID-UNITS-PER-BOX           TO NA-NUM.
MOVE 0                          TO NA-DECIMAL-LEN.
CALL "NUM-TO-TEXT"              USING NUM-AREA.
MOVE NA-TEXT                    TO XML-OUT-ATR-VALUE (35).
MOVE "Backorder-sw"             TO XML-OUT-ATR-LABEL (36).
MOVE ID-BACKORDER-SW            TO XML-OUT-ATR-VALUE (36).
MOVE "Bill-of-material-sw"      TO XML-OUT-ATR-LABEL (37).
MOVE ID-BILL-OF-MATERIAL-SW     TO XML-OUT-ATR-VALUE (37).
MOVE "Catalog-item-sw"          TO XML-OUT-ATR-LABEL (38).
MOVE ID-CATALOG-ITEM-SW         TO XML-OUT-ATR-VALUE (38).
MOVE "Start-date"               TO XML-OUT-ATR-LABEL (39).
MOVE " "                        TO WS-TIME.
CALL "X-DATE-TIME-STAMP" USING GLOB ID-START-DATE
                           WS-TIME XML-OUT-ATR-VALUE (39).
MOVE "Dimension-width"          TO XML-OUT-ATR-LABEL (40).
MOVE ID-DIMENSION-WIDTH         TO NA-NUM.
CALL "NUM-TO-TEXT"              USING NUM-AREA.
MOVE NA-TEXT                    TO XML-OUT-ATR-VALUE (40).

MOVE "Dimension-length"         TO XML-OUT-ATR-LABEL (41).
MOVE ID-DIMENSION-LENGTH        TO NA-NUM.
CALL "NUM-TO-TEXT"              USING NUM-AREA.
MOVE NA-TEXT                    TO XML-OUT-ATR-VALUE (41).
MOVE "Dimension-height"         TO XML-OUT-ATR-LABEL (42).
MOVE ID-DIMENSION-HEIGHT        TO NA-NUM.
CALL "NUM-TO-TEXT"              USING NUM-AREA.
MOVE NA-TEXT                    TO XML-OUT-ATR-VALUE (42).
MOVE "Special-handling-sw"      TO XML-OUT-ATR-LABEL (43).
MOVE ID-SPECIAL-HANDLING-SW     TO XML-OUT-ATR-VALUE (43).
MOVE "Taxable-sw"               TO XML-OUT-ATR-LABEL (44).
MOVE ID-TAXABLE-SW              TO XML-OUT-ATR-VALUE (44).
MOVE "Free-space-width"         TO XML-OUT-ATR-LABEL (45).
MOVE ID-FREE-SPACE-WIDTH        TO NA-NUM.
```

```
           CALL "NUM-TO-TEXT"              USING NUM-AREA.
           MOVE NA-TEXT                    TO XML-OUT-ATR-VALUE (45).
           MOVE "Free-space-length"        TO XML-OUT-ATR-LABEL (46).
           MOVE ID-FREE-SPACE-LENGTH       TO NA-NUM.
           CALL "NUM-TO-TEXT"              USING NUM-AREA.
           MOVE NA-TEXT                    TO XML-OUT-ATR-VALUE (46).
           MOVE "Free-space-height"        TO XML-OUT-ATR-LABEL (47).
           MOVE ID-FREE-SPACE-HEIGHT       TO NA-NUM.
           CALL "NUM-TO-TEXT"              USING NUM-AREA.
           MOVE NA-TEXT                    TO XML-OUT-ATR-VALUE (47).
           MOVE "Reverse-kit"              TO XML-OUT-ATR-LABEL (48).
           MOVE ID-REVERSE-KIT             TO XML-OUT-ATR-VALUE (48).
           MOVE "Max-weight"               TO XML-OUT-ATR-LABEL (49).
           MOVE ID-MAX-WEIGHT              TO NA-NUM.
           CALL "NUM-TO-TEXT"              USING NUM-AREA.
           MOVE NA-TEXT                    TO XML-OUT-ATR-VALUE (49).
           MOVE "Discontinued-sw"          TO XML-OUT-ATR-LABEL (50).
           MOVE ID-DISCONTINUED-SW         TO XML-OUT-ATR-VALUE (50).

           MOVE "Max-line-qty"             TO XML-OUT-ATR-LABEL (51).
           MOVE ID-MAX-LINE-QTY            TO NA-NUM.
           MOVE 0                          TO NA-DECIMAL-LEN.
           CALL "NUM-TO-TEXT"              USING NUM-AREA.
           MOVE NA-TEXT                    TO XML-OUT-ATR-VALUE (51).
           MOVE "Non-machine-sw"           TO XML-OUT-ATR-LABEL (52).
           MOVE ID-NON-MACHINE-SW          TO XML-OUT-ATR-VALUE (52).
           MOVE "Insert-prod-1"            TO XML-OUT-ATR-LABEL (53).
           MOVE ID-INSERT-PROD-1           TO XML-OUT-ATR-VALUE (53).
           MOVE "Insert-prod-2"            TO XML-OUT-ATR-LABEL (54).
           MOVE ID-INSERT-PROD-2           TO XML-OUT-ATR-VALUE (54).
           MOVE "Max-order-qty"            TO XML-OUT-ATR-LABEL (55).
           MOVE ID-MAX-ORDER-QTY           TO NA-NUM.
           MOVE 0                          TO NA-DECIMAL-LEN.
           CALL "NUM-TO-TEXT"              USING NUM-AREA.
           MOVE NA-TEXT                    TO XML-OUT-ATR-VALUE (55).
           MOVE "Lot-control-sw"           TO XML-OUT-ATR-LABEL (56).
           MOVE ID-LOT-CONTROL-SW          TO XML-OUT-ATR-VALUE (56).

           MOVE 56                         TO XML-OUT-AT-NO.
           CALL "X-OUT-ADD-ELEMENT" USING GLOB XML-OUT XML-OUT-BUF.

       END PROGRAM CS-GETITEM.
/
```