

**Subject:** PEC IVR Inter-Machine Call  
Transfers using Dialogic HMP  
firmware

**Parwan Electronics Corporation**  
[www.voicesaver.com](http://www.voicesaver.com)  
Tel. (732)-290-1900 x 3100  
February 7, 2019  
No. 2019-0204

## TECHNICAL MEMORANDUM

### Description

The PEC Insight™ IVR Call Transfer block supports the blind transfer of a call from one machine running HMP to another machine using HMP. Both machines must be connected to Cisco or other voice gateways that route call to the machines running PEC software. Cisco must be provisioned to route the call to the extension specified in the Number to Dial field of the IVR Call Transfer block.

### IVR Call Transfer from the Transferred Machine

The IVR Call Transfer block must be set as follows:

Field	Value
Label of Phone Number	
Call Transfer Prefix	
Transfer Type [a / b / c]	a
Label of mail box	
Connect Label	
No Answer Label	
Busy Label	
Answering Machine Label	
Intercept Label	
Disconnect Line Label	
New Call Label	
Label of File to play	
Number to Dial	1999999999@198.168.0.6
Name of File to play	
Call Transfer Suffix	
Tear Reason Label	
DBname (3chs)(label)	
Agent Port (label)	

Once the IVR Call Transfer block is processed the software transfers the call to the address specified in the Number to Dial field of the block. Options the address may be specified in the Label. Along with the extension and IP address, the the software sends the five XML tag variables to the transferee machine. In order to the do the software send the “Referred-By: [sip:1@1.1.1.1;%s](http://sip:1@1.1.1.1;%s)” to Cisco, where %s corresponds the XML tag data. In the example above 1999999999 is the extension number and 198.168.0.6 is the IP

address of the transferred machine. Make sure Cisco routes the call to the transferee machine, Please note your XML tag variable data should not have any blanks in them.

### ***Handling the Call on the Transferee Machine.***

The transferee machine receives the call in the mailbox associated with the extension of the Number of Dial of the IVR Call Transfer block. The mailbox on the Transferee machine must be an E type mailbox. The software reads the Cisco message received and move the XML tag data into corresponding variables. Cisco sends the “Requested-By: [sip:1@1.1.1.1;%s](mailto:sip:1@1.1.1.1;%s)” message. The IVR logic can use the XML variables to do the further processing. The following image shows how the mailbox must be set:

The screenshot displays the 'Box Setup' configuration window. At the top, there is a navigation bar with icons for 'gs', 'Sys Info', 'M.Boxes', 'Tree', 'IVR', 'SysProp', and 'Notes'. The main area is titled 'Box Setup' and contains a form with the following fields and values:

- Type: E, Class: (empty), Options: (empty)
- Box Number: 1999999999
- Box Name: ?????XXXX
- Passcode: 2222
- Max Messages: 100
- Longest Msg: 120
- IVR File Path: C:\STAS\_2900\CONFIVR
- Pager Type: 1
- Pager Code: 2
- Retry Count: 1
- Pager DND: (unchecked)
- Email Code: (empty)

Summary statistics are shown in the top right:

- New Messages: 3
- Total Messages: 3
- Calls: 100
- Minutes: 9
- Beeps: 0
- RecID: 5

Additional settings include: Rings to Wait, Supervised Transfer Option (1), Credit [mins] (0), Call Screen, Company, Greeting Play Mode (0), Domain, PBX Intercom Page Zone, Wake Up Time, Notification Time, Escalation Box, Voice Recognition, Sampling Rate, Forward to, Hold Msgs for (empty) Days, Msg Light On, and Msg Light Off.

The bottom toolbar contains buttons: F10 - Save, Exit, Del Msgs, Del Int, Clear Calls, Default, F2 - Find, Add New, Previous, Next, and More.

In the example above, the call is received in the 1999999999 mailbox and the IVR in the folder c:\stas\_2900\confivr further processes the call.

## Cisco Programming

Cisco must be programmed to route the call to the proper machine. Before the data is sent to Cisco, the software converts the data for Cisco. The following C code is used:

```
#include <stdio.h>
#include <ctype.h>

char rfc3986[256] = {0};
char html5[256] = {0};

void encode( const char *s, char *enc, char *tb)
{
    for (; *s; s++)
    {
        if (tb[*s])
        {
            sprintf(enc, "%c", tb[*s]);
        }
        else
        {
            sprintf(enc, "%%%02X", *s);
        }
        while (++enc);
    }
}

Void
PEC_Transferred_From_Machine( char *zXml_Var, char *zData_to_Cisco )
{
    char *url ; // ORIGINAL XML DATA
    char *enc ; // ENCODED FOR CISCO

    url = zXml_Var Data;
    enc = zData_For_Cisco;

    int i;
    for (i = 0; i < 256; i++)
    {
        rfc3986[i] = isalnum(i) || i == '~' || i == '-' || i == '.' || i ==
        ? i : 0;
        html5[i] = isalnum(i) || i == '*' || i == '-' || i == '.' || i ==
        ? i : (i == ' ') ? '+' : 0;
    }

    encode(url, enc, rfc3986);
    puts(enc);

    return 0;
}
```

PEC Confidential Information

1230 Highway 34, Aberdeen, NJ 07747 – USA – [www.voicesaver.com](http://www.voicesaver.com) - Phone: 732-290-1900

Page 3/6

When the software received the data from Cisco, it decodes the data and move it to the XML tag variables. The following C code is used to do the decoding:

```
#include <stdio.h>
#include <string.h>

Void
PEC_Transferred_To_Machine( char *zXml_Data, char *zData_From_Cisco )
{
    decode( zData_From_Cisco, zXml_Data );
}

inline int ishex(int x)
{
    return (x >= '0' && x <= '9') ||
           (x >= 'a' && x <= 'f') ||
           (x >= 'A' && x <= 'F');
}

int decode(const char *s, char *dec)
{
    char *o;
    const char *end = s + strlen(s);
    int c;

    for (o = dec; s <= end; o++) {
        c = *s++;
        if (c == '+') c = ' ';
        else if (c == '%' && (!ishex(*s++) ||
                               !ishex(*s++) ||
                               !sscanf(s - 2, "%2x", &c)))
            return -1;

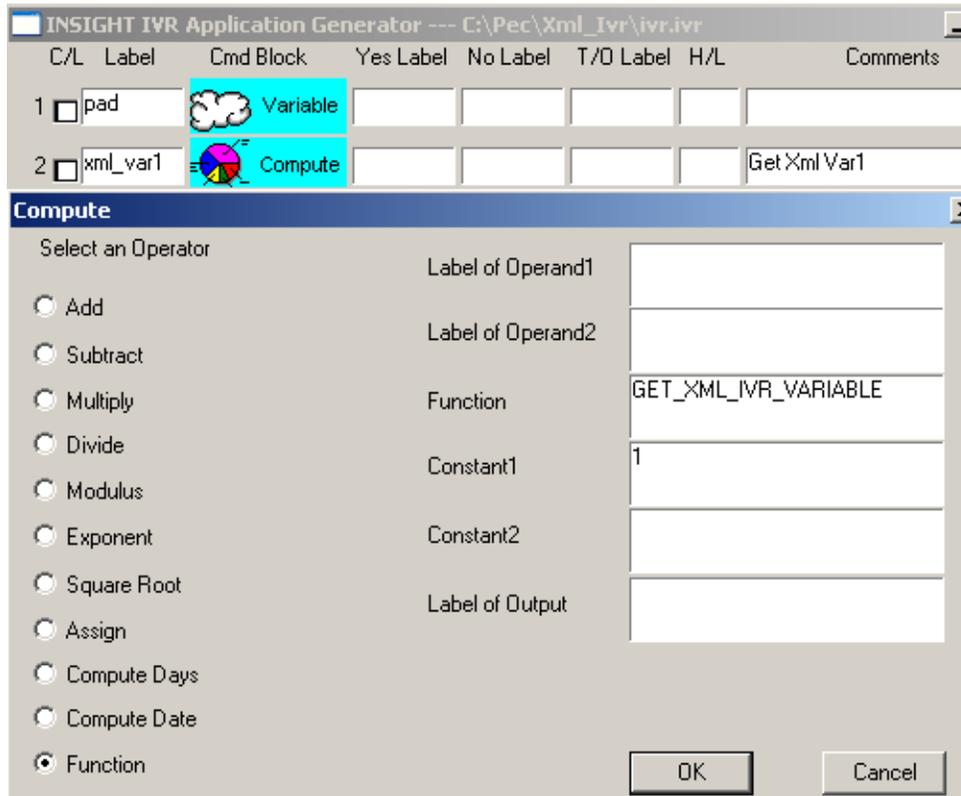
        if (dec) *o = c;
    }

    return o - dec;
}
```

DRAFT FEB. 8, 2019

## Reading the Xml Tag Variables by the IVR

The IVR Applications Developer must use the Compute block of the Applications Generator to get the Xml tag variables into a label variable. Please see the following example:



The above compute block will get the Xml tag Var1 to the xml\_var1 label. In order to get the Xml tag Var2, set the Constan1 to 2.

DRAFT

## Setting the Xml Tag Variables from the IVR

The IVR Applications Developer must use the Compute block of the Applications Generator to set the Xml tag variables. Please see the following example:

Select an Operator	Label of Operand1	
<input type="radio"/> Add	Label of Operand2	xml_var1
<input type="radio"/> Subtract	Function	SET_XML_IVR_VARIABLE
<input type="radio"/> Multiply	Constant1	1
<input type="radio"/> Divide	Constant2	
<input type="radio"/> Modulus	Label of Output	
<input type="radio"/> Exponent		
<input type="radio"/> Square Root		
<input type="radio"/> Assign		
<input type="radio"/> Compute Days		
<input type="radio"/> Compute Date		
<input checked="" type="radio"/> Function		

OK Cancel

The above compute block sets the Xml tag var1 to the data in the xml\_var1 label. To set the Xml tag var2, then set Constant1 to 2.

DRAFT FEB 19