Host (on behalf of ASD):

ADS is the Premier Trade Organisation for companies in the UK Aerospace, Defence, Security and Space Sectors.

Condition Based Maintenance (CBM+) with Highly Interactive Electronic Technical Manuals (IETM)

Integrating the Process DM External Application Interface

Name of presenter: Robert Pearson

Rank/title of presenter: Senior Logistics Analyst

Company/organization: Pentecom, LLC

S1000D User Forum, London *October 14-16, 2019*

Presenter Biography

Bob Pearson is a principal subject matter expert in the implementation of advanced interactive functionality in S1000D for Pentecom, LLC.

He is a former chairman of the S1000D United States Standards Implementation Group (USSIG) and emeritus member of the Electronics Publications Working Group (EPWG), as well as having been an active contributor to the United States Standards Management Group (USSMG) Sea Working Group.

Bob spent 36 years with Raytheon and predecessor company Hughes Aircraft serving many roles in logistics and technical publications. He worked closely with the ongoing effort to utilize GIEA-STD-007 Logistic Product Data directly in S1000D publications.

Bob graduated with a Bachelor of Science in Electrical and Electronics Engineering (BS-EEE) from NDSU in 1982.

Condition Based Maintenance and S1000D

Defense organizations today are facing an environment that demands improved performance at reduced cost. One of the promising approaches to meet these goals is condition based maintenance (CBM). CBM implementations have been shown to improve maintainer accuracy, efficiency, and speed while also allowing for enterprise consolidation and reuse of the data - further reducing costs.

Condition Based Maintenance and S1000D

S1000D provides a standards based, open method to implement condition based maintenance that allows the troubleshooting and repair actions to be driven by the current state of the system.

In addition to the traditional technical manual information, such as Illustrated Parts Breakdowns, Descriptive Information, Procedures, etc., S1000D has Processes.

Processes allow subject matter experts to capture the exact action flow needed to respond to each of the unique failure mode conditions of a system and to guide the maintainer through the precise steps needed for that exact situation.

IETM capability differences - Traditional vs Highly Interactive

- S1000D IETMs always provide the functionality that traditional paper/PDF tech manuals provided. Procedures and Descriptive Content have limited advanced features such as the possibility of hot-spotted graphics and linking, and in general have the following characteristics:
  - Static, Linear data organization.
  - End user determined navigation through the material
  - Material can be printed as a complete manual or procedure.
  - Have no means of direct interface with other systems - rely on End User knowledge to properly preform the task at hand.

Traditional example

…

25. Look at the fault readout on the Health Maintenance Console (HMC).
26. Choose the fault code shown from the list below.
    a) FFAC: Go to Step 105.
    b) FFBD: Go to Step 132.
    c) FFBF: Go to Step 210.
    d) ----: Go to Step 27.
27. Cycle power on the HMC.

…

Traditional vs Highly Interactive differences

- The term Highly Interactive generally refers to advanced Interactive Electronic Technical Manuals (IETMs) that have self contained condition based navigation and the capability to interface directly with the maintainer and/or other systems.
  - Non-linear structured data (similar to software programming constructs).
  - Navigation is programmed into the data and determined in real time by the current conditions in the environment.
  - Processes cannot be printed in advance as a whole since they are dependent upon input conditions.
  - Can accept input, such as fault conditions, directly from the maintainer and/or other systems.

Highly Interactive Example

…

Gathering Fault Information. Press NEXT.

< Maintainer selects NEXT >

< IETM communicates with the HMS and retrieves fault code(s)>

< IETM Navigates to the corrective action based on the fault, in this case a failed network switch.>

Cycle power on Network Switch A. Press Next when complete.

<Maintainer selects NEXT>

<IETM communicates with HMS and retrieves fault codes>

<No fault reported, cycle power cleared fault>

Fault Corrected. Press NEXT to end.

What is the Process Data Module?

- The Process Data Module captures programming logic in the XML data itself.
  - It allows a system subject matter expert to record the exact sequence of actions needed for each combination of environmental and configuration conditions.
  - The appropriate set of actions is determined in real time based on input from the Maintainer, and/or the system itself.
  - Uses standard programming constructs:
    - Execution Sequences.
    - If-Then-Else structures
    - Alternative Selection structures
    - Loop structures
    - Dialogs
    - External Process calls

Why use Processes?

- Faster, more accurate troubleshooting/repair.
- Correct execution of actions determined by subject matter experts.
- Less system specific training required for the maintainer - intelligence about the system is captured in the data.
- Guided navigation reduces end user errors - no need to jump around and keep track of multiple tasks at once.
- Input directly from the system under test can be used to automatically present the appropriate corrective action without relying on maintainer best judgment.

How does a Process Navigate?

- Processes have a fixed navigation sequence with decision points that are affected by system conditions and maintainer input.
- The maintainer uses NEXT and PREVIOUS functions to move forward and backward in the sequence.
- Each “Stop” along the sequence can:
  - Display another data module.
  - Display a dialog to get input from the maintainer
  - Contain sets of steps that are not part of a different data module.
  - Collect input from an External Application.

How does a Process Navigate? General Node Processing

Operator presses NEXT function.

Filtering on Next Node evaluated based on current system conditions.

If the filtering conditions are met, any optional entry conditions are set and processing continues.

The content is displayed.

Dialogs or External Applications can change the system state.

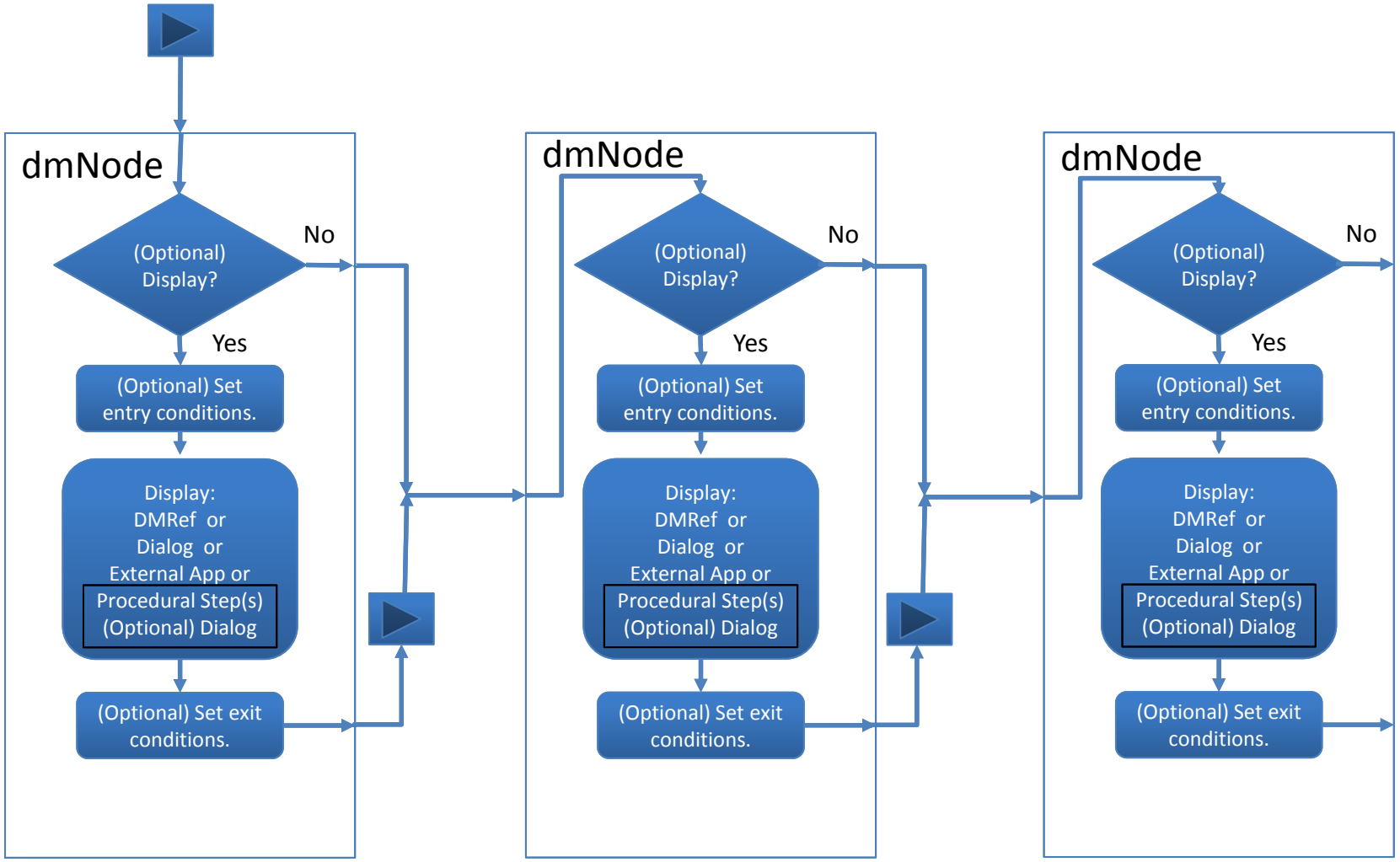
Operator makes selections, if any, then presses NEXT.

Any optional exit conditions defined by the author are executed, altering the system state.

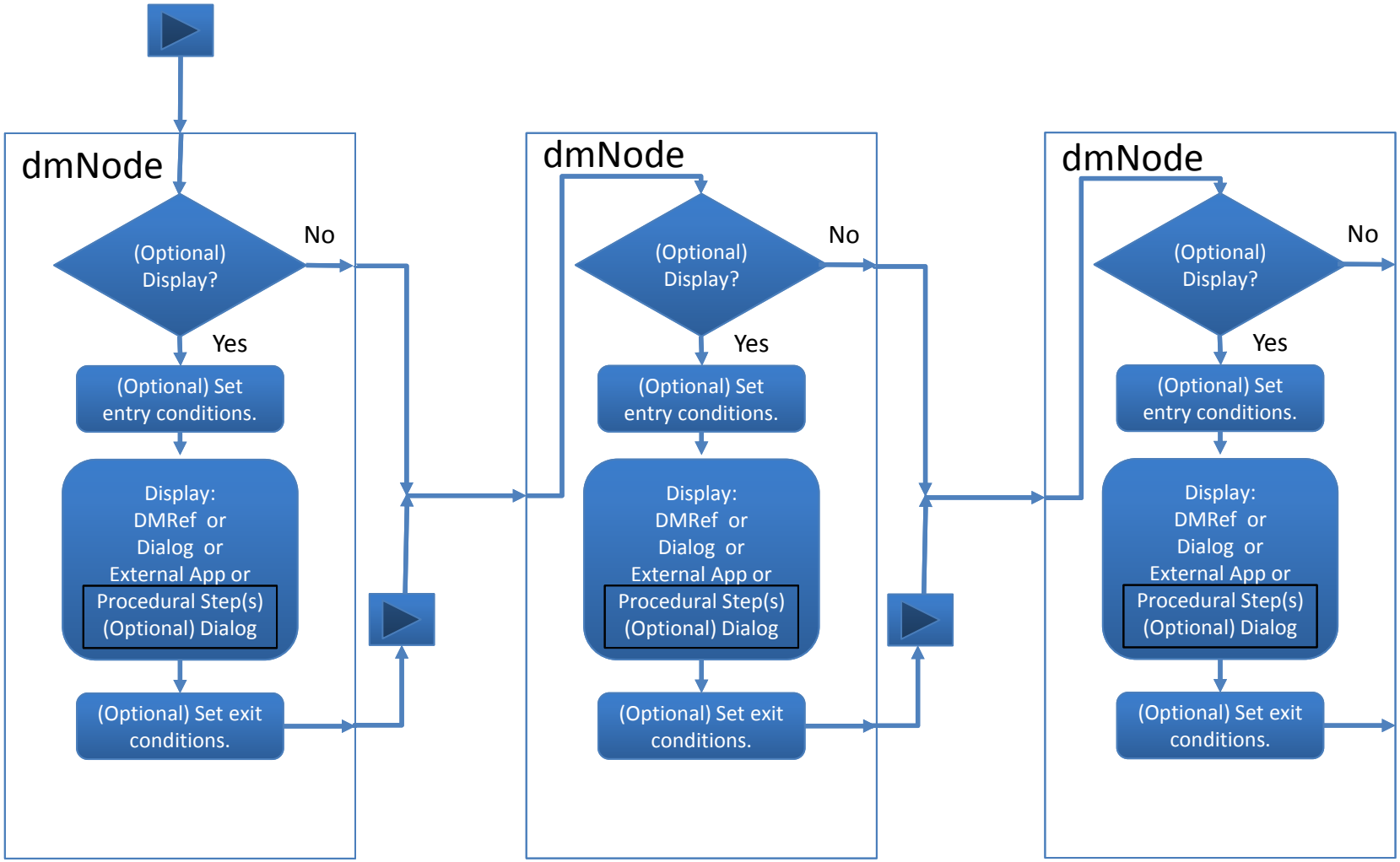
Filtering on the Next Node is evaluated against the new system state.

Continue….

How does a Process Navigate? General Node Processing



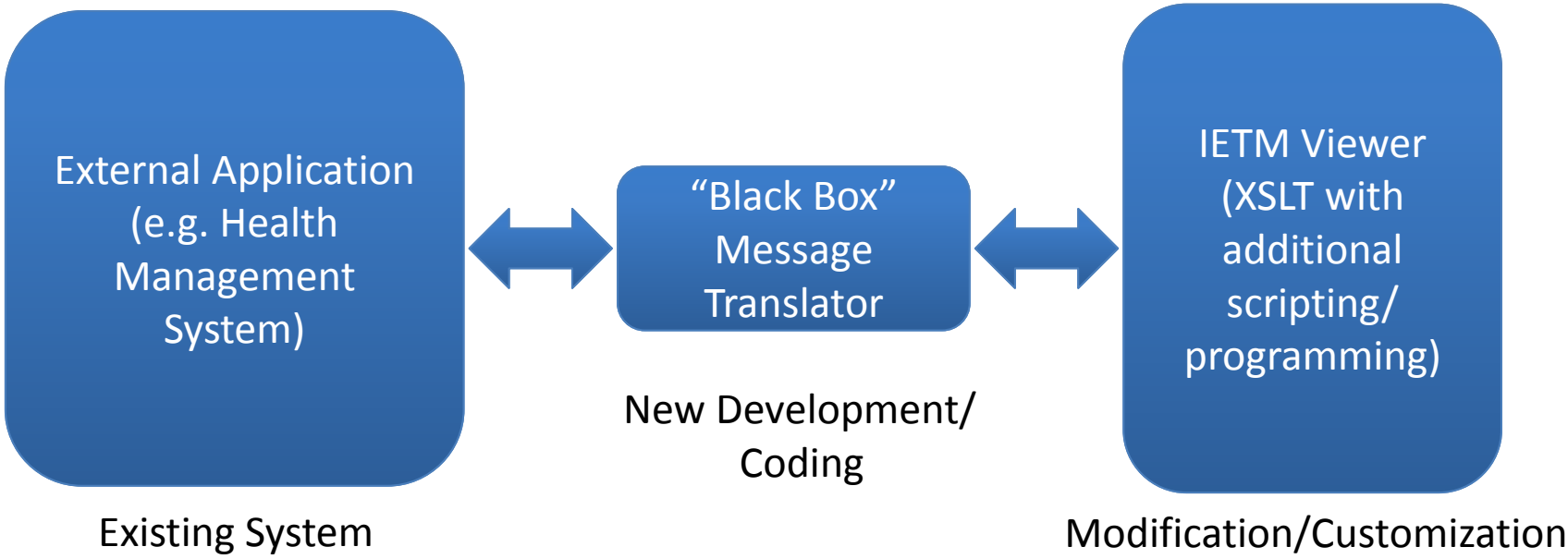
How does a Process Navigate? Sequence Example



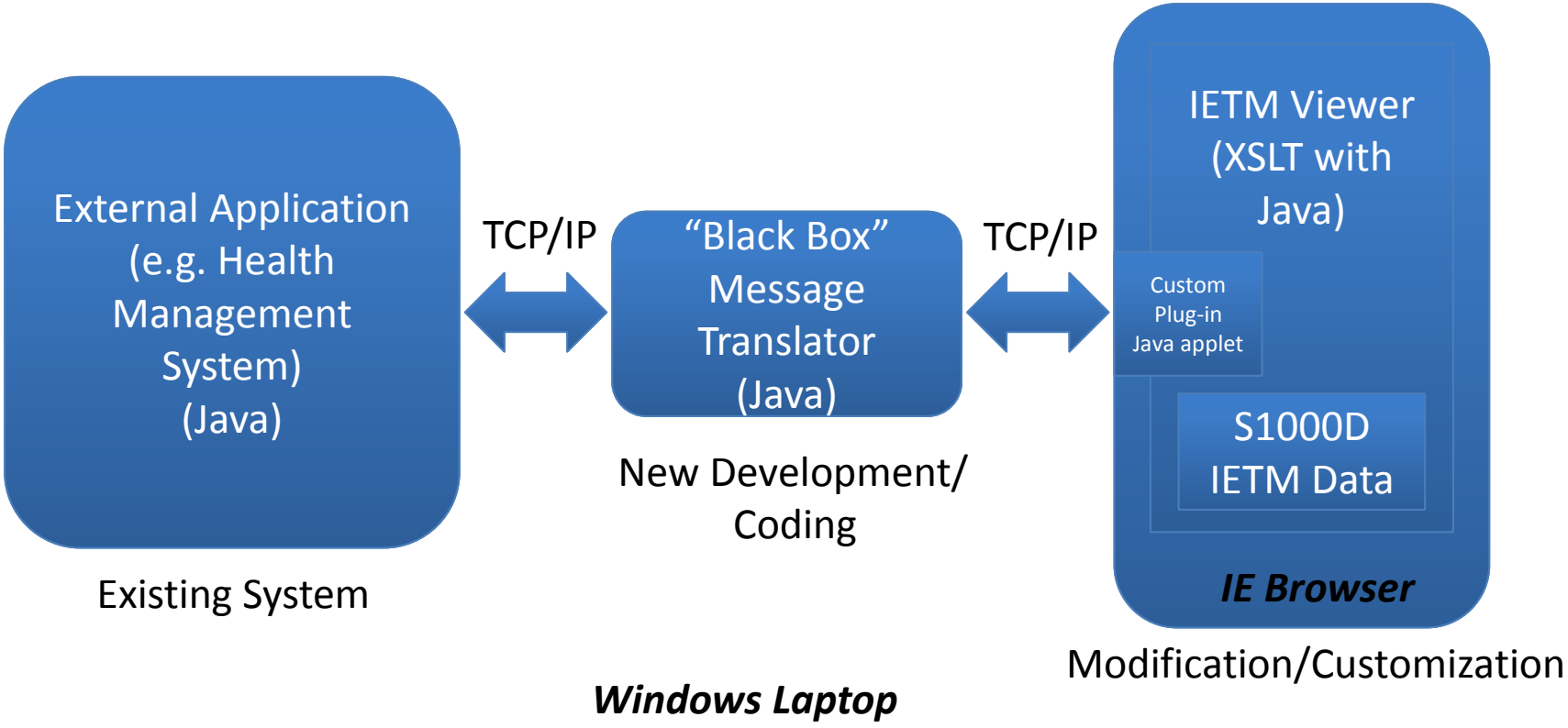
How can you troubleshoot based on current system conditions?

- The Process Data Module has a programming construct called the External Application.
- The External Application interface is generic and can be implemented using any technology or programming language.
- Communication with the External Application, (such as the host system or a Health Maintenance System) is achieved by designing and coding a “Black Box” that is unique to the specific host system and messages that need to be translated to/from S1000D External Application XML.

Main Components of an Integrated Solution



Example implementation, co-resident on the same platform



Example of an External Application Message

```
<dmNode>
<!--Retrieve sysUnit, sysCabinet, sysDevice, and sysFaultCode from the bbtranslator.-->
    <externalApplication application="host.bbtranslator.main.hms">
        <paraBasic>Retrieve HMS parameters</paraBasic>
        <send>
            <sendName>method</sendName>
            <stringValue>getHMSParams</stringValue>
        </send>
```

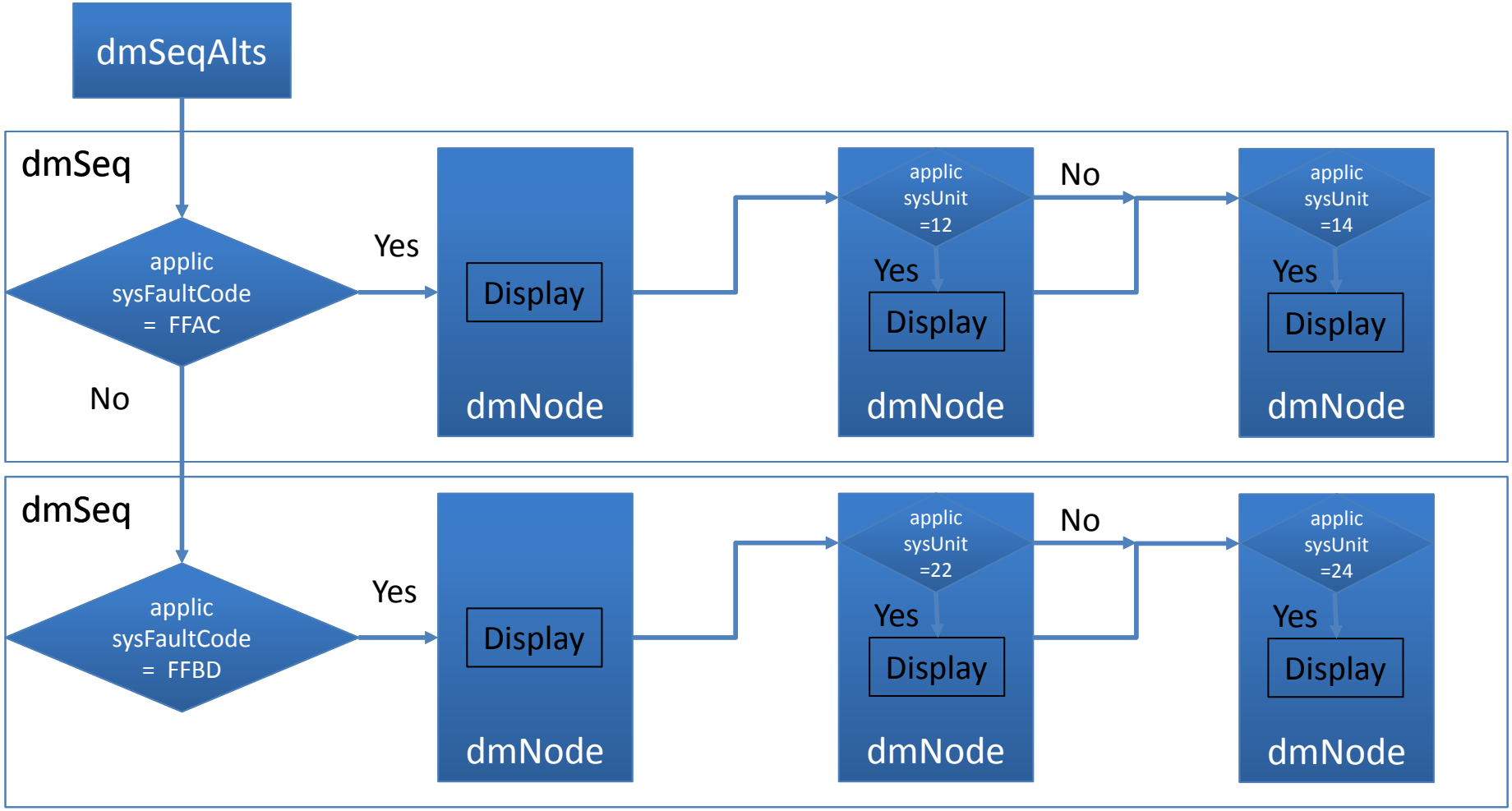
Example of an External Application Message (cont'd)

```
<receiveByName>
    <returnedValueName>sysUnit</returnedValueName>
    <globalPropertyRef applicPropertyType="condition"
    applicPropertyIdent="sysUnit" />
</receiveByName>
<receiveByName>
    <returnedValueName>sysCabinet</returnedValueNam
    e>
    <globalPropertyRef applicPropertyType="condition"
    applicPropertyIdent="sysCabinet" />
</receiveByName>
```

Example of an External Application Message (cont'd)

```
        <receiveByName>
                <returndValueName>sysDevice</returnedValueName>
                <globalPropertyRef applicPropertyType="condition"
                applicPropertyIdent="sysDeviceId" />
        </receiveByName>
        <receiveByName>
                <returnedValueName>sysFaultCode</returnedValueName>
        <globalPropertyRef applicPropertyType="condition"
                        applicPropertyIdent="sysFaultCode" />
        </receiveByName>
    </externalApplication>
</dmNode>
```

Troubleshooting is filtered by the information received.



Troubleshooting is filtered by the information received.

- In the previous example, the first sequence is executed for fault FFAC, which may be a Power Supply Fault.
  - The first dmNode tells the maintainer the nature of the failure and the suspect failed unit.
  - The second dmNode is only displayed for unit 12, which is power supply A, and links to the Remove/Install for that unit.
  - The third dmNode is only displayed for unit 14, which is power supply B, and links to the Remove/Install for that unit.
- The result is that the maintainer will only see the steps that are appropriate for the corrective action that is specific to the exact fault and faulted unit.

Thank you

for your attention!

Questions?