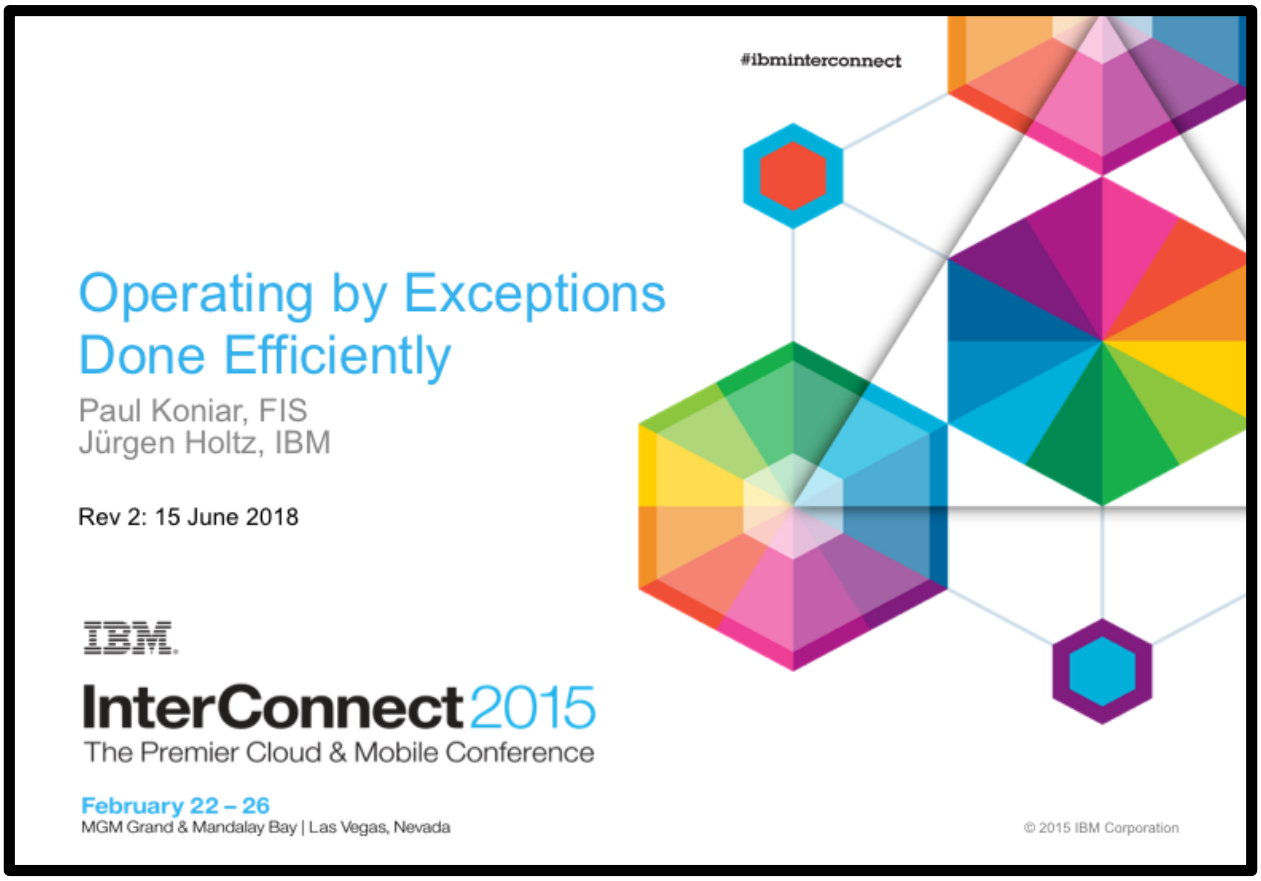
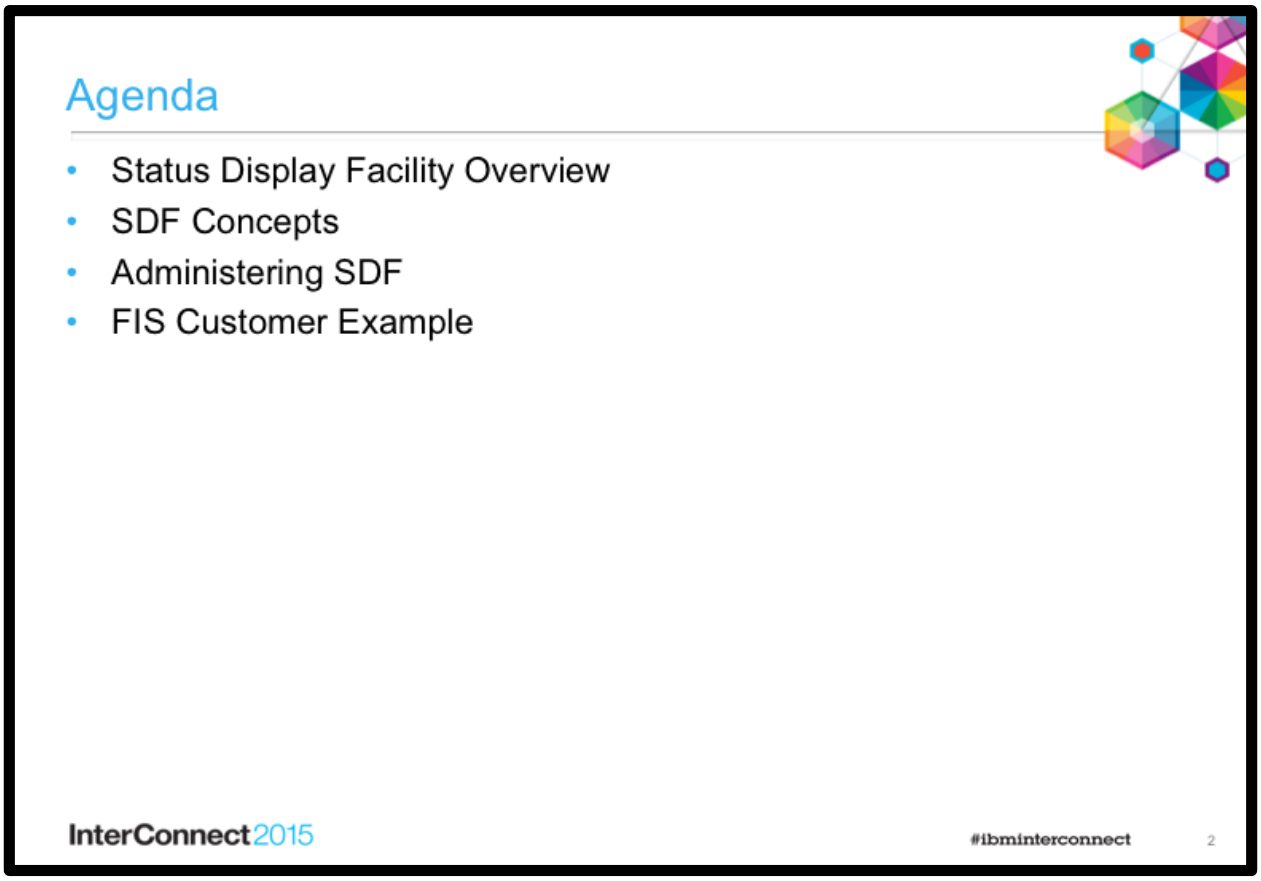


Are you drowned by too much data that doesn't allow you to quickly spot what is going wrong? Then, this session is just the thing for you! Enterprises have large amounts of operational status information that need to be aggregated, sorted, and presented in a very fast, robust, reliable and at the same time flexible way.

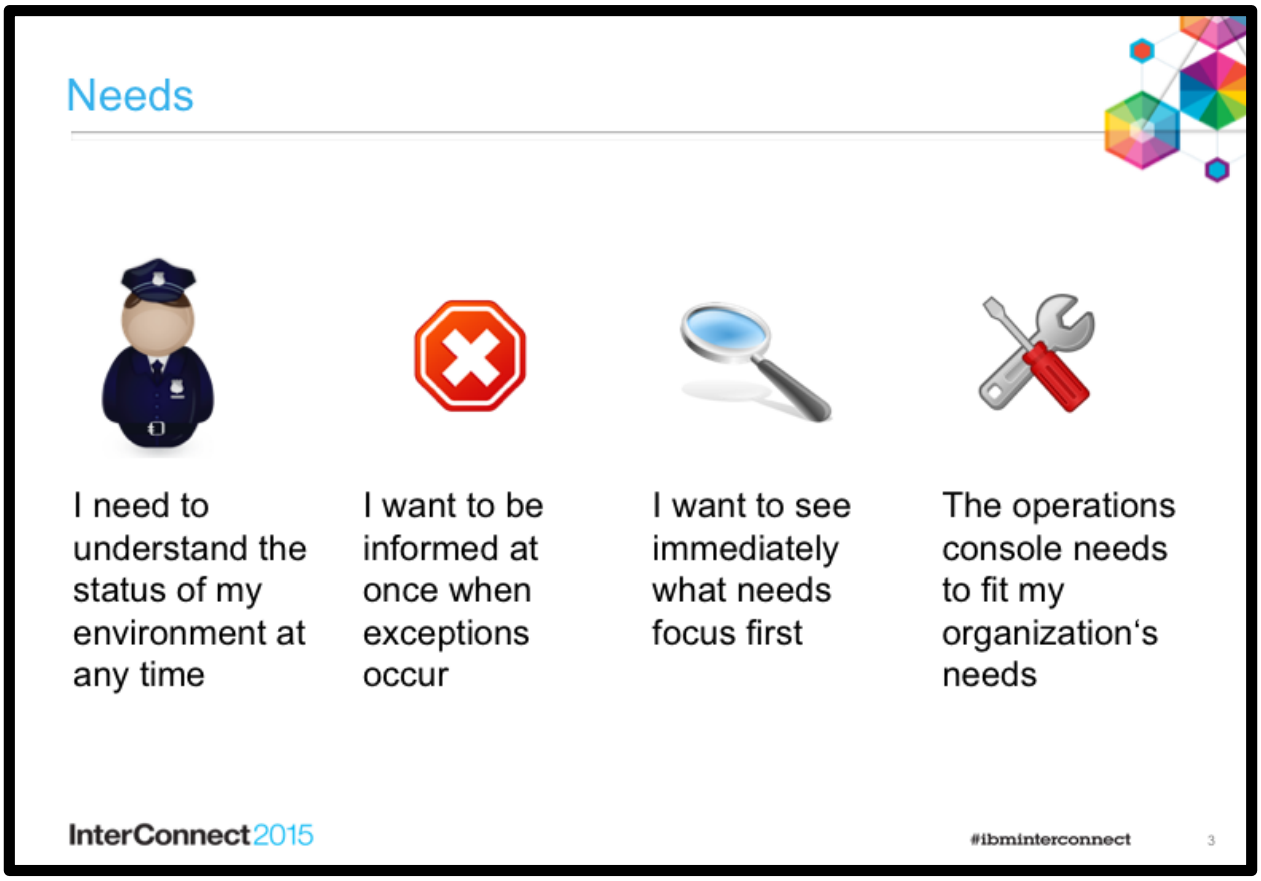
In this session, the speakers will introduce SA z/OS's Status Display Facility which does just that. You will see how to use this facility as shipped out-of-the-box. In addition you will learn how to customize the tool based on a specific customer environment, to unleash the power of it and to supervise your environment very efficiently from a single point of control.



In the first part of this presentation, an introduction of the Status Display Facility (SDF) will be given. You will see what characteristics are considered most with respect to an efficient operating console and how SDF addresses each of them. A brief overview of the capabilities that come out-of-the-box is provided as well.

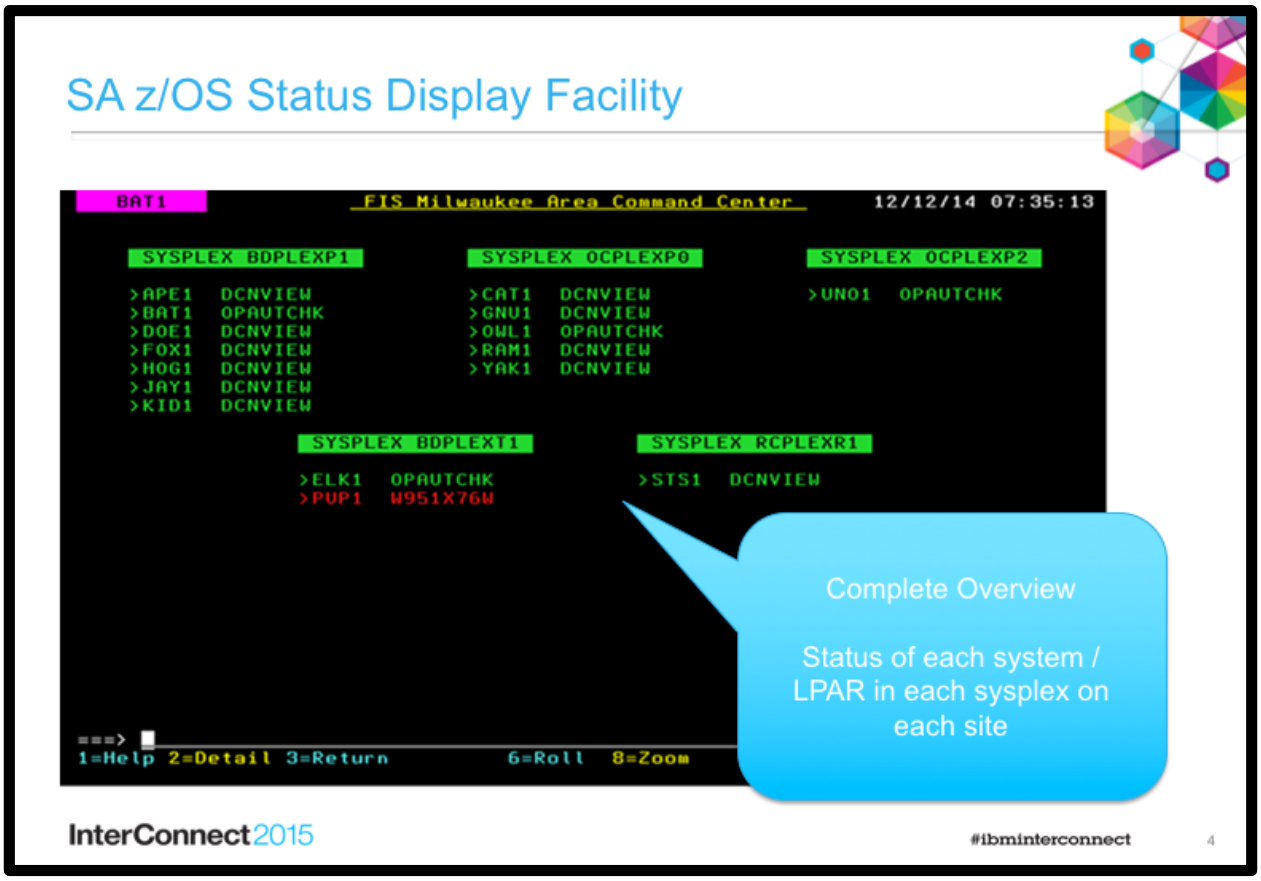
The second part focuses on the concepts. Terms like *Status Descriptor, Status Component* and *Status Tree* are introduced. This section also shows you how to define panels and use them to present the status to your operators. More advanced topics such as the recently introduced major components, exits and administrative commands are touched as well.

Finally, you will see how SDF is deployed at Fidelity Investment Services (FIS). You see how a large environment can effectively operated with just a few panels accommodated specificially to the needs of this installation.

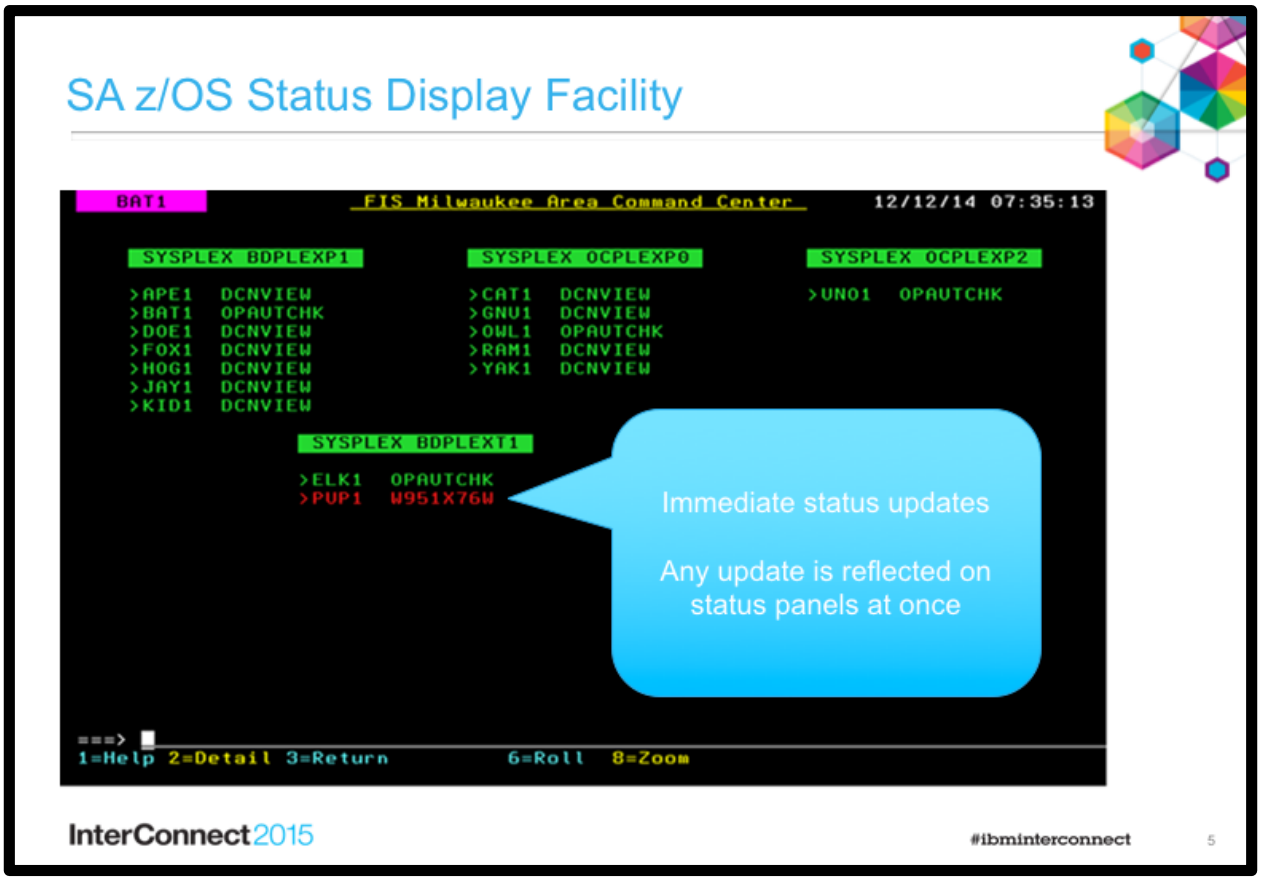


What are the typical needs to operate efficiently?

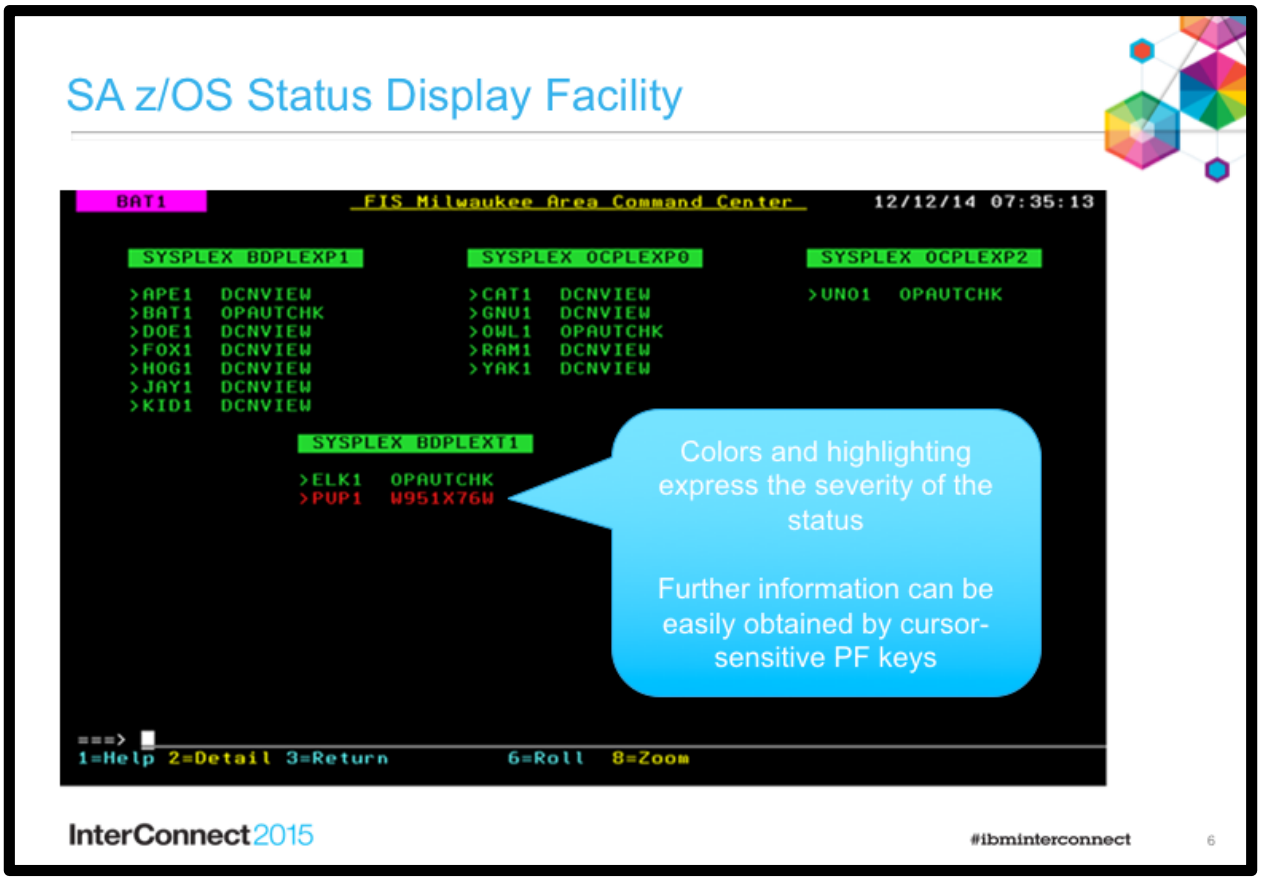
1. I need to understand the status of every item that matters to my operations in the entire environment at any time. I am looking for answers for questions such as: 'Are the critical applications up and running?', 'Is anybody blocked by outstanding operator replies?' or 'Are the systems and its components in a healthy condition?'
2. Whenever something exceptional happens, for example an application failed and cannot be recovered, I would like to see this status being immediately reported to me.
3. In a large enterprise, there can be tens of unusual events at the same time. I want to see immediately what the most critical events are and hence need focus first.
4. Finally, no installation is like another. To meet the needs for my operators, the operations console must be customizable such that it fits my organization's needs.



SDF allows you to design your panels such that you can oversee your whole IBM Z environment. In fact, SDF allows you to structure the panels in an hierarchical order that lead you from overviews into more detailed views. Typical overviews group systems or LPARs by sysplex and site. Normal status is typically shown in green. Absence of any status can be indicated in another color to distinguish this situation from normal or exceptional situations. Any status obtained on a particular system will be reflected immediately on the local SDF panels. When an SDF focal point is defined (what is usually the case), the status will also be immediately forwarded to the focal point. At any time, the operator knows whether the environment is operating well or not.



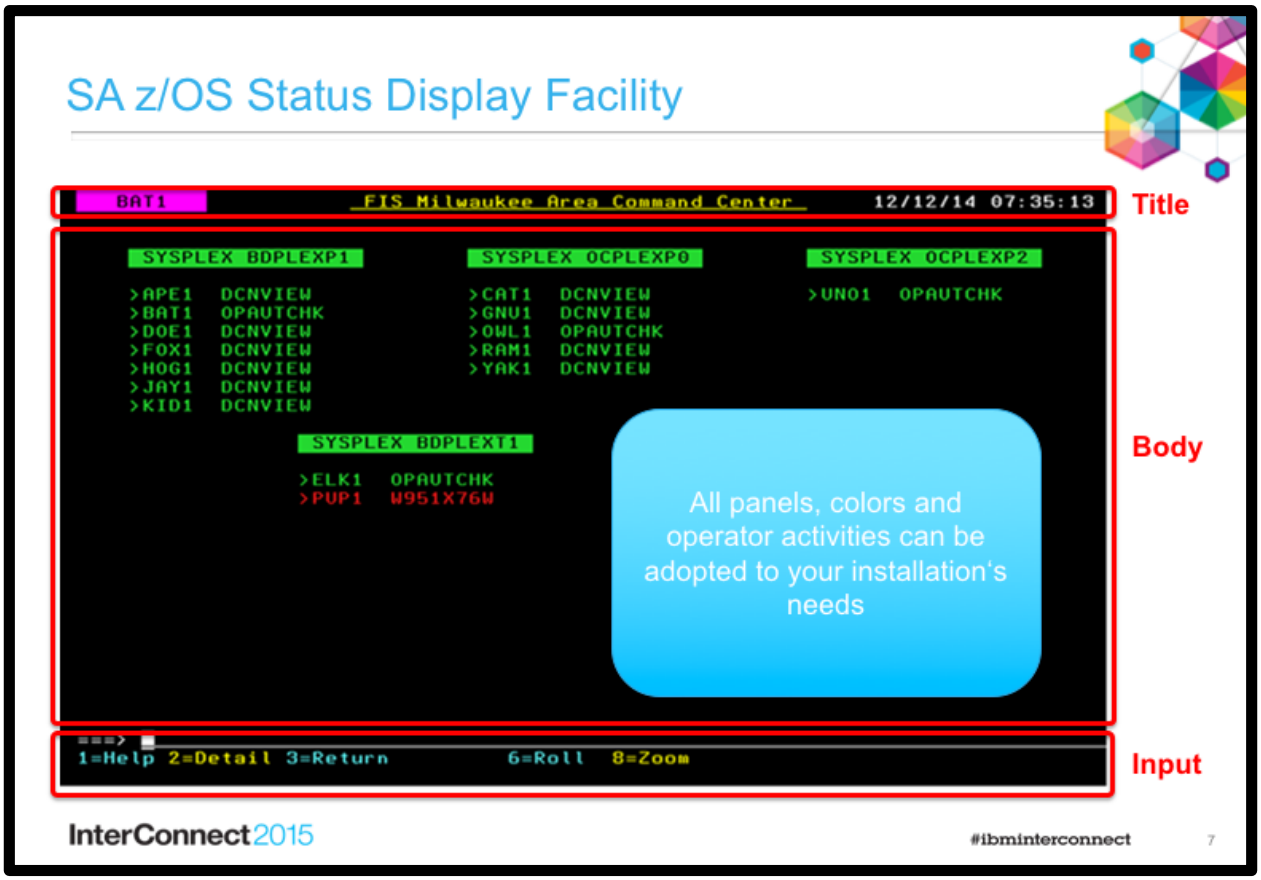
When any status changes on any of the systems, it is reflected on the status panel at once. The use of colors and highlighting allows you to visually distinguish critical status values from less critical status values or from informational status values. In total there are 28 different ways to visually represent a particular status for the operator resulting as the combination of 7 colors with 4 highlighting schemes.



Once you have identified a particular *status field* of interest, operators can place the cursor over the colored text and press any of the programmable function (PF) keys for more information. For example, on some panels, pressing the key PF8 (“Zoom” / “Down”) will display a more detailed status panel that shows different status categories like subsystems and critical messages on behalf of the status field where the cursor was located. The operator can then see which underlying status was propagated to the previous panel. Status panels can be ordered hierarchically to enable further drill down or navigation to specific views.

Detailed information about a status field can be obtained by placing the cursor on it and pressing the key PF2 (“Detail”). This leads to a chain of one to many detail panels that the operator can flip through in priority order. Higher priority status details are displayed before lower priority status details are shown until the end of the chain is reached.

Detail panels are not further discussed in this presentation. Their layout and content is fixed and cannot be changed by the customer. When *Status Descriptors* are discussed later on, you will understand what information you see on a Detail panel.

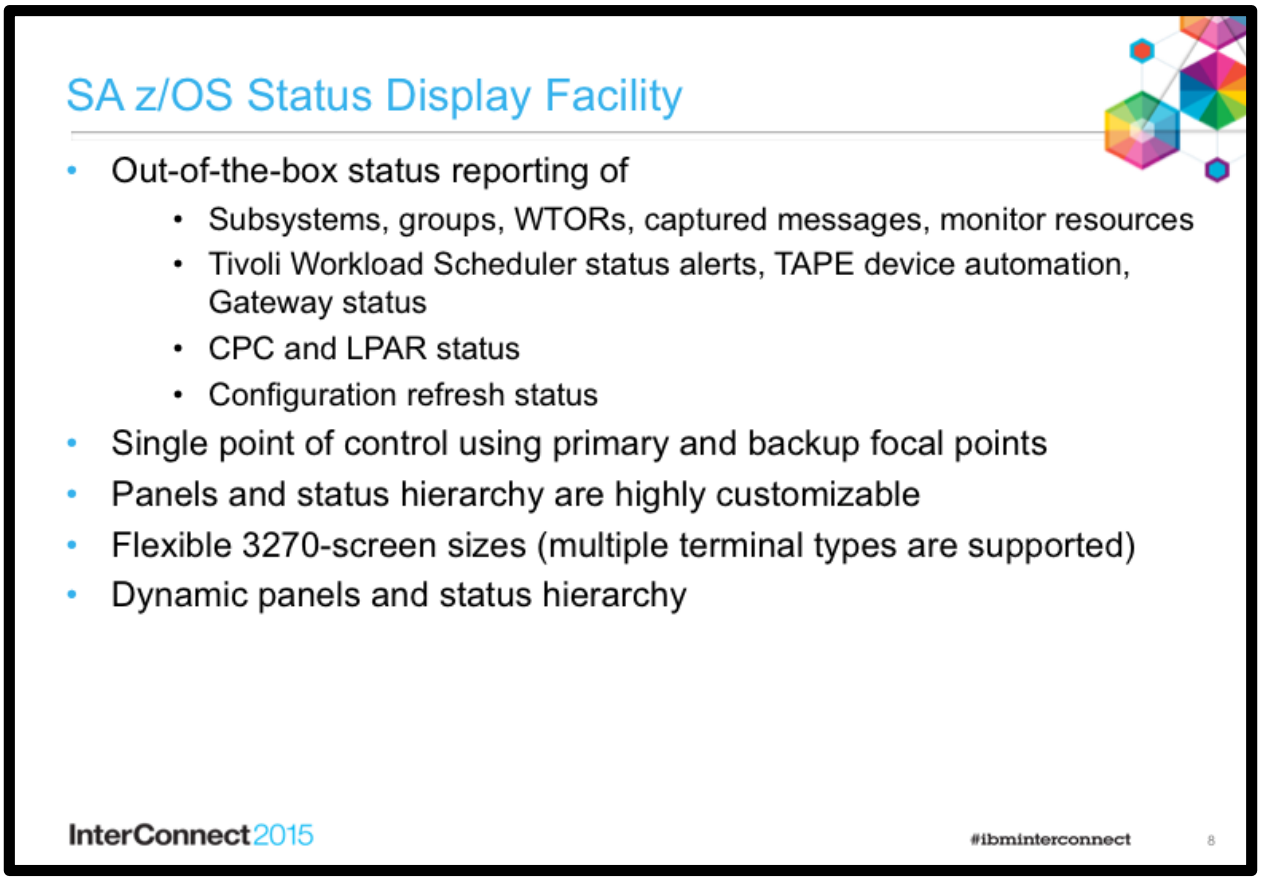


SDF is highly flexible. SA z/OS comes with a default *status tree* and a set of default *panels* that allow you to operate your standard environment, both locally on each system but also from a central focal point. But every installation can modify this status tree or even create their own trees to define a status hierarchy that best matches the installation's needs.

Similarly, panels can be modified or designed completely anew to meet the requirements of the operators. This allows operators to find the most important information in a compact form on a single screen. It also allows them to perform further problem isolation and to act on critical status only by using PF keys.

Each panel has a *title* area, a *body* area and an *input* area with PF keys. This structure is not enforced by any panel definition statement but merely a good coding practice. *Text fields* are used to display static content on the panel in each of these areas.

The body area additionally consists of *status fields*, one or more scrollable *bodies* or a combination of status fields and bodies. A body in turn is a tabular display of status information, that can be replicated over two or more columns up to the right side of the physical screen. This allows you to take advantage of large screen sizes with minimal panel definition efforts.



When System Automation for z/OS is installed, SDF brings many capabilities already out of the box. The sample definitions that come with the product allow you to monitor

- Subsystems, i.e. APL type of resources
- Groups, i.e. APG type of resources
- Write-to-Operator-with-Reply messages that are not answered yet
- Messages that have been captured based on policy definitions or using AOFCPMSG
- Health status, i.e. MTR type of resources
- Tivoli Workload Scheduler status alerts
- Status information about tape device automation
- Gateways, i.e. the domain-to-domain connectivity between different SA z/OS systems

You can also monitor the status of

- Processors and
- LPARs

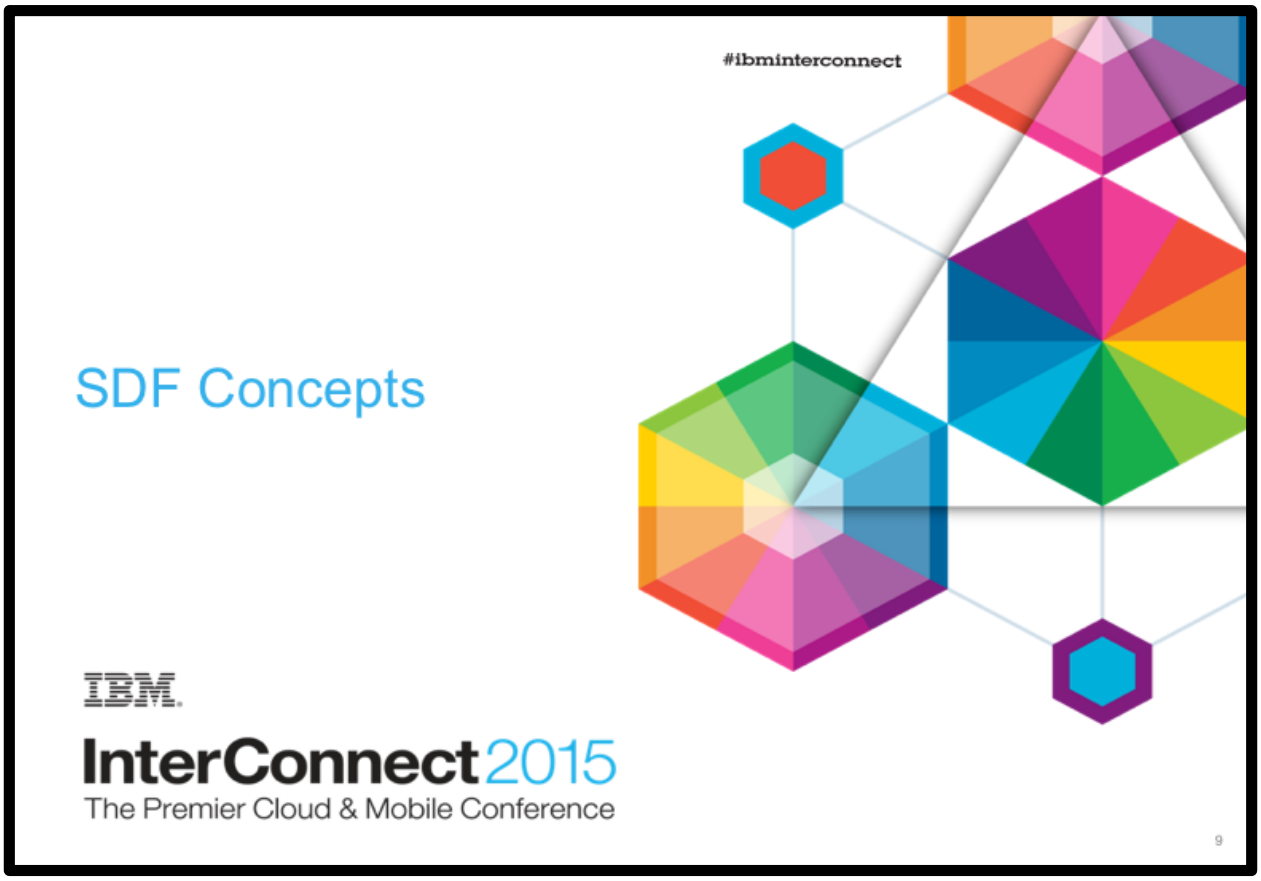
Above status is provided by System Automation for z/OS Processor Operations

(ProcOps) and will be automatically included on the ProcOps focal point system.

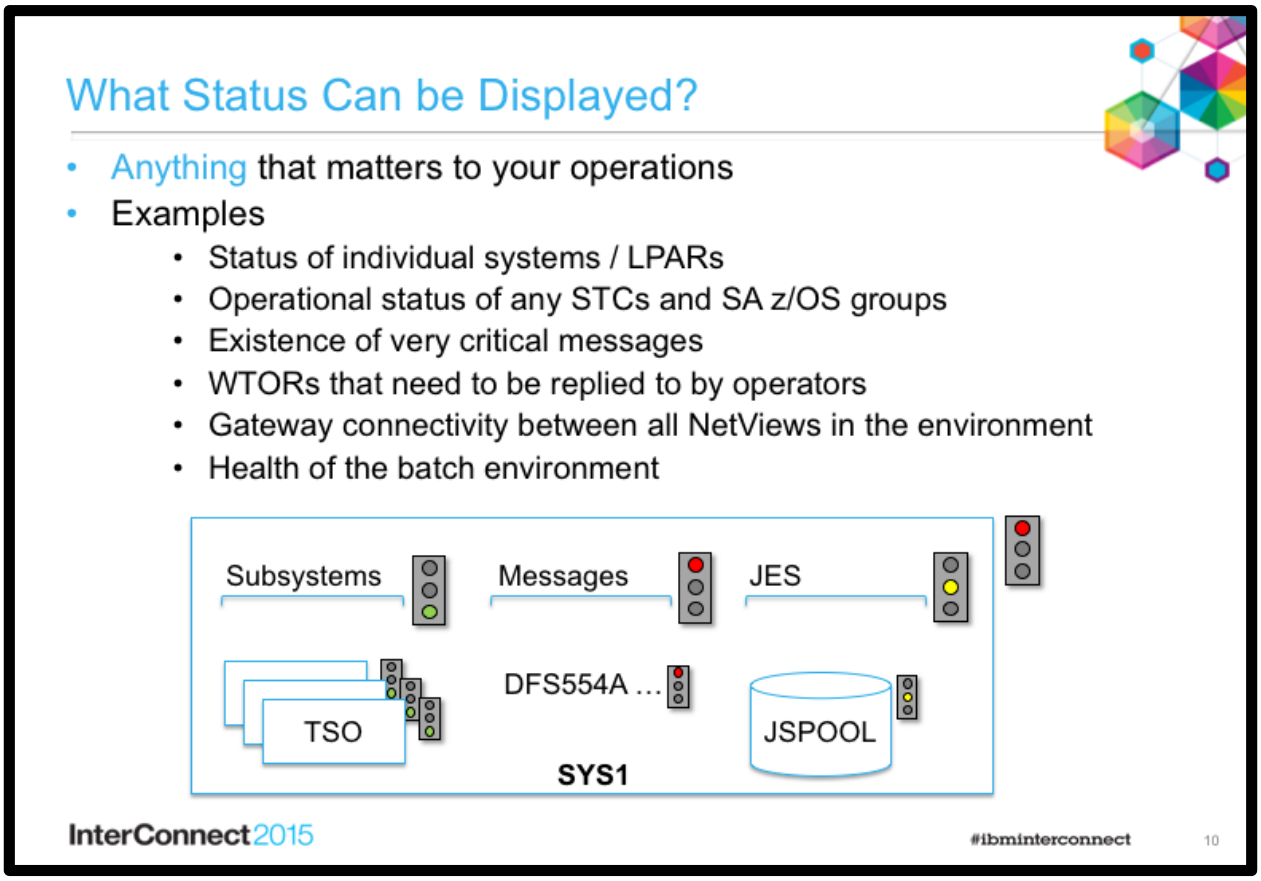
Since SA z/OS V3.5, you can monitor the progress of automation policy refreshes within your enterprise.

The hierarchy of your status data and the layout and content of the panels can be changed to fit the organisation's needs, if the samples are not sufficient. In particular dynamic panels and status trees allow you to design panels with minimal effort for your entire enterprise.

SDF supports the typical terminal models, for example 3278 mod 2/3/4/5. Actually, you can design the panels with any screen size you like as long as they are not less than 80 or more than 160 characters wide and not less than 24 or more than 62 characters high.



The following pages explain the underlying concepts of SDF. In a nutshell, you will understand what kind of status can be displayed, how you can update the status of a particular object, how SDF supports you to aggregate status within a hierarchy and what capabilities SDF brings to display aggregated or detailed status on a 3270-terminal to best meet operators' needs.



Starting from scratch, you first have to ask yourself, what status matters to your operations. Objects that come to mind, immediately are system and LPAR status and the status of applications (STCs, i.e. resources of type APL) as well as of application groups (APGs).

Operators often need to be alerted when important, exceptional messages are issued or when others wait for WTOR replies. So these too affect the operational status.

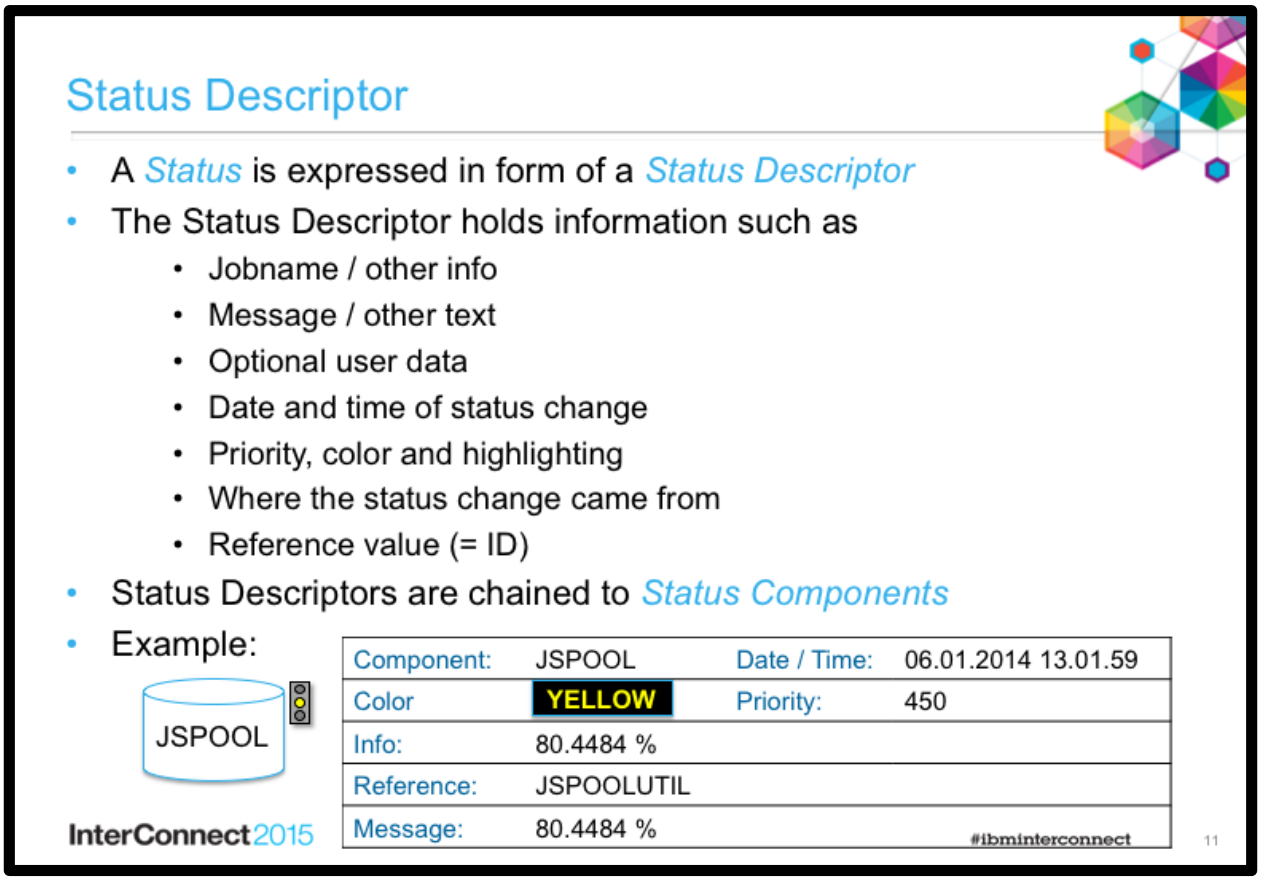
In a large environment, organisations rely on a central operations console concept (SPOC = single point of contact). When the communication pathes between the systems do not work properly, this is also a measure of degraded operational status. Gateway communication is the means used by System Automation to exchange information between the systems across sysplex boundaries.

In many cases, a healthy batch environment is equally important than the online transaction processing workloads. Understanding the utilization of critical JES resources like SPOOL therefore is key.

Such objects can be categorized. Individual address spaces can be grouped under subsystems and individual messages can be grouped into a messages category. The status of the next higher category reflects the worst status of its contained objects. In case of batch, the SPOOL can be one object whose status is grouped under a category called JES together with the status of free job numbers or free output queue

elements. The overall system status in turn is built by finding the worst status in all its underlying categories.

But how are these objects defined in SDF and how is status reported? This will be explained on the subsequent pages.



Status is expressed in form of a *Status Descriptor* that holds a pre-defined set of attributes. Historically, these attributes are aligned closely to System Automation's resources, like subsystems and messages. Later on, when other resource types have been introduced, the existing attributes have been reused.

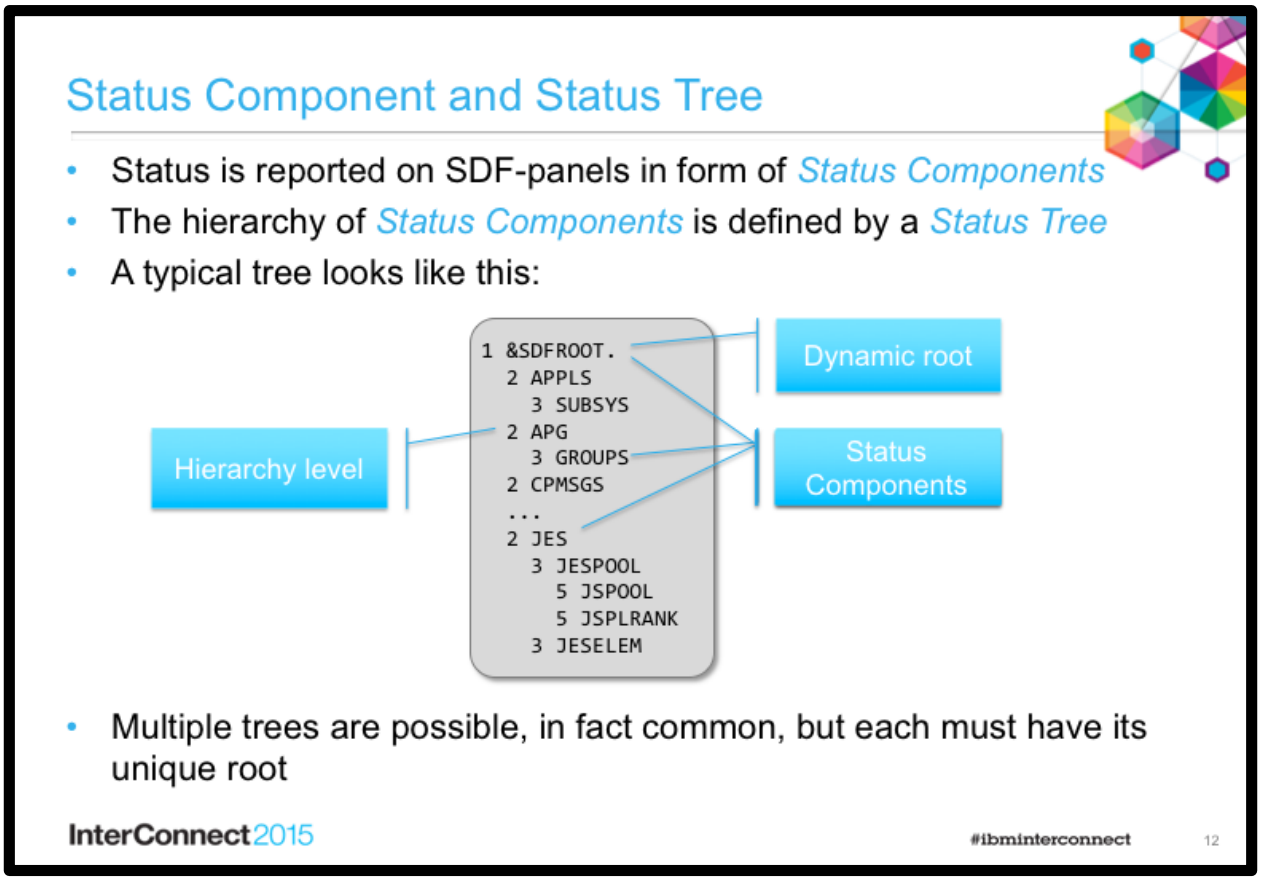
The attributes that exist today are:

- Jobname or Info. The Detail panel will present this as Jobname, when the Status Descriptor is chained to the SUBSYS component. Otherwise, this attribute will be presented as Info. In the latter case, it is arbitrary text.
- Message. This is also arbitrary text, usually a message associated with the status, for example AOF571I reporting the new agent status of a subsystem.
- 1..240 bytes of arbitrary user data.
- Date and time when the Status Descriptor was created.
- The priority of the descriptor. The priorities range from 1 to 99999999. The lower the number, the higher its priority.
- The display color of the descriptor. Typically, you chose a color that is aligned with the priority of the descriptor. This attribute is ignored, when a color is specified on the panel in a body.
- The display highlighting of the descriptor. Typically, you chose a highlight that is

aligned with the priority of the descriptor. This attribute is ignored, when a highlight is specified on the panel in a status field or in a body.

- The reference value of the descriptor. This acts as an ID and is important when an old descriptor needs to be replaced by a new descriptor, when the status has changed or to get rid of a descriptor, when it is no longer needed.
- The origin of the descriptor, i.e. the NetView auto-operator that drove the status update request (AOCUPDT or SA-internal).
- System name and NetView domain of the origin.

Every Status Descriptor is chained to one or more *Status Components*, discussed next.



To report status, a *Status Component* is needed. The Status Component determines the level of granularity being reported on SDF panels. Status Components are hierarchically structured in a so-called *Status Tree*. In a typical SDF-environment, there are multiple trees, one for each system, for instance.

A level-1 element is a status component also called *root*. The roots must be unique. Subordinate levels can be created as needed and with level numbers as you like as long as you obey to the AOFTREE syntax rules described in Chapter 6. SDF Definition Statements in the Programmer's Reference manual.

A status tree that contains the &SDFROOT. variable is a dynamic tree. All roots are defined in a single NetView common global stem variable called AOF_AAO_SDFROOT.. During initialization time, SDF creates one status tree for every token found in this variable. Hence, you can define one single tree and let SDF generate as many unique trees as you have systems in your environment.

The alternative to a dynamic tree is a static tree. In this case, SDF doesn't touch the tree during initialization. More on that topic later on.

On the course of this presentation, we will create a status monitor for important JES resources. Therefore, the default status tree that comes with the product has been expanded as shown in the picture above. JES is the Status Component that aggregates all JES-specific status. Underneath, there is a JESPOOL Status Component

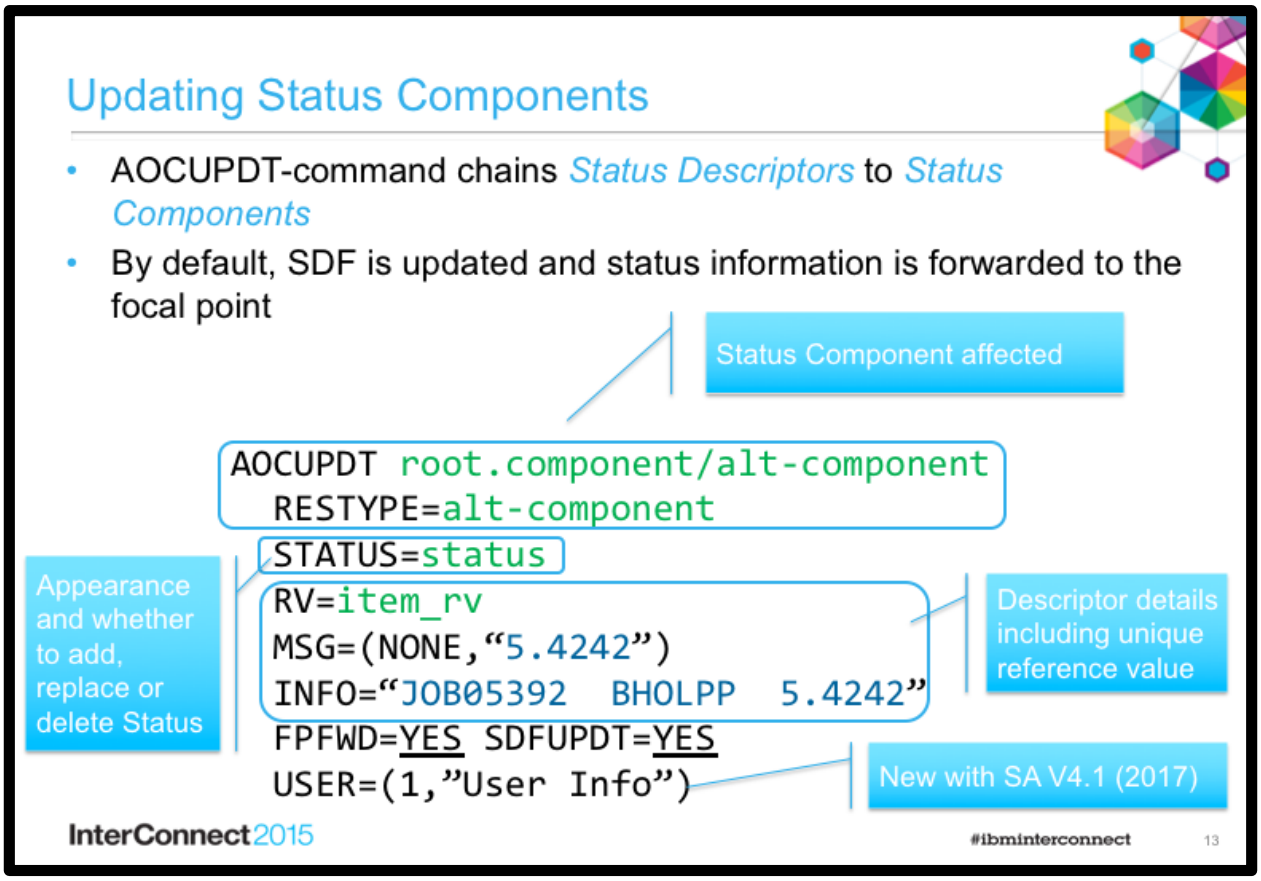
that merely groups the two components JSPOOL and JSPLRANK. JSPOOL represents the SPOOL utilization. The idea is that its message field reports the actual SPOOL utilization. The status will change from normal to warning, once the utilization is higher than the system-defined warning threshold. You will see later on, how "normal" or "warning" are mapped to a status.

There is also the Status Component, JSPLRANK, which represents the aggregated status of a top-n list of jobs or started tasks using space on the SPOOL. The sample monitor (code not discussed in this presentation) will chain Status Descriptors using names JSPLRANK1 to JSPLRANK15 for the top 15 users at this Status Component (JSPLRANK).

Finally, there is the JESELEM Status Component which represents the aggregated status of the four important JES resources

1. Free job numbers
2. Free job queue elements
3. Free job output elements
4. Free BIRTs (block extension reuse table)

When the warning threshold for any of these JES resources is exceeded, the status will change from normal to warning.

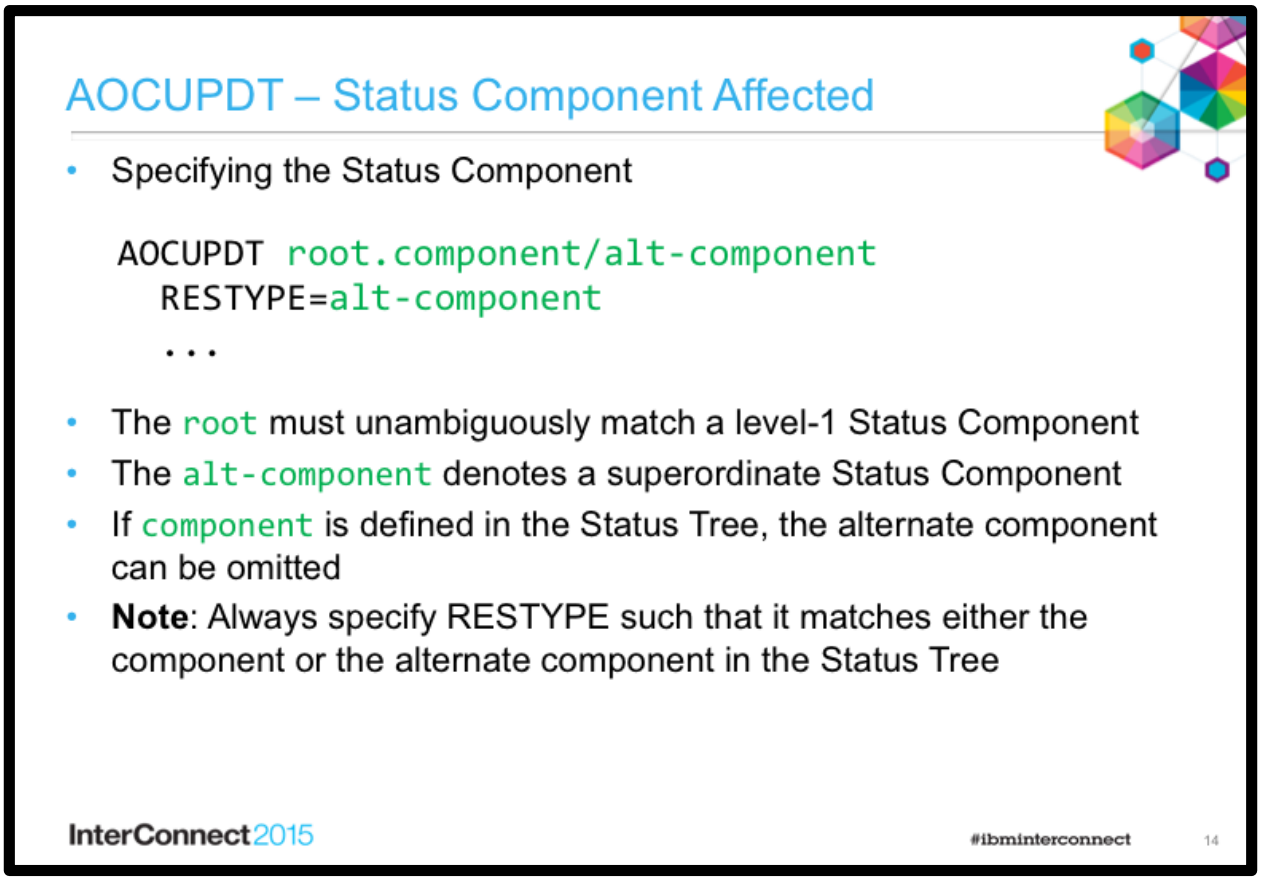


To update Status Components, a new Status Descriptor needs to be created that either is chained to this Status Component, that replaces one or more existing descriptors chained already to this component, or that actually causes the removal of matching descriptor(s) from this component.

To chain a Status Descriptor to a component, the AOCUPDT-command is used. When nothing else is specified, the default parameters ensure that SDF is updated (you need to understand that AOCUPDT is a multi-purpose update function) and that the status update is also forwarded to the SDF focal point.

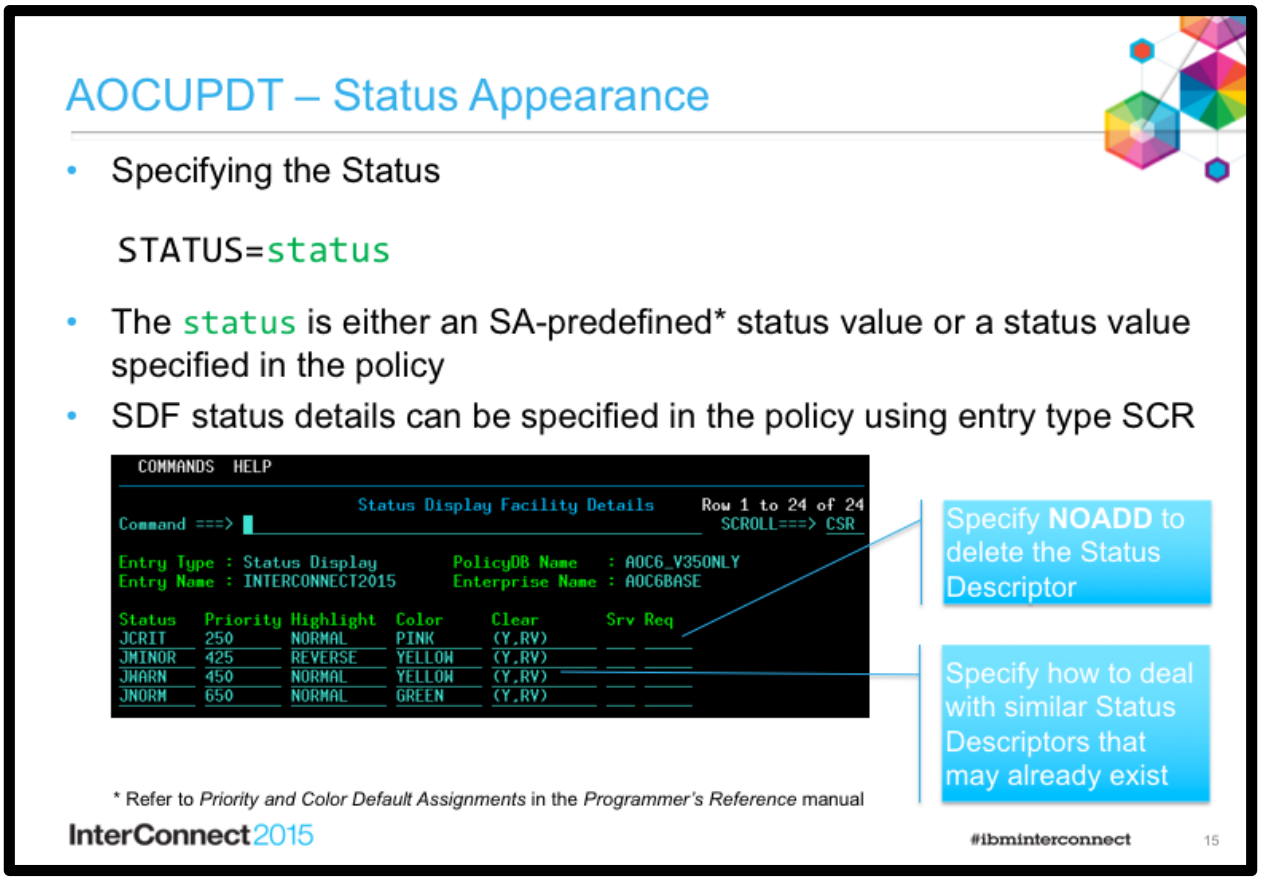
The descriptor attributes stem from the RV (reference value), MSG (message) and INFO parameters. Priority, color and highlighting will be described in a bit. The next page describes how the Status Component is specified to which this descriptor is chained.

Note that since SA z/OS V4.1, the AOCUPDT-command also accepts user data that can be passed using the USER-keyword. Up to 240 bytes are available in a buffer and information can be placed anywhere in that buffer. Later on, the user data can be fetched from the buffer for display or your own PF-keys using the STATUSFIELD or CELL statements (described further below).



You have to specify the (unique) `root` such that SDF knows into which tree the `component` is added. A message, an STC, every object is a component, whether it is defined in the tree or not. If you choose, for example, an STC like TSO and you have not also defined TSO as a Status Component in the tree, then you append the superordinate component TSO belongs to as the alternate component (`alt-component`) separated by a slash. In this case, it is /SUBSYS (pre-defined for SA z/OS use). However, had you defined TSO in the tree, the alternate component can be omitted.

For programmers a bit confusing is the fact that you also have to specify the RESTYPE parameter. It has no particular purpose other than to ensure that the value of this parameter matches one of the defined Status Components in the tree. For good practice, specify the alternate component name or if it exists, the name of the `component` itself.

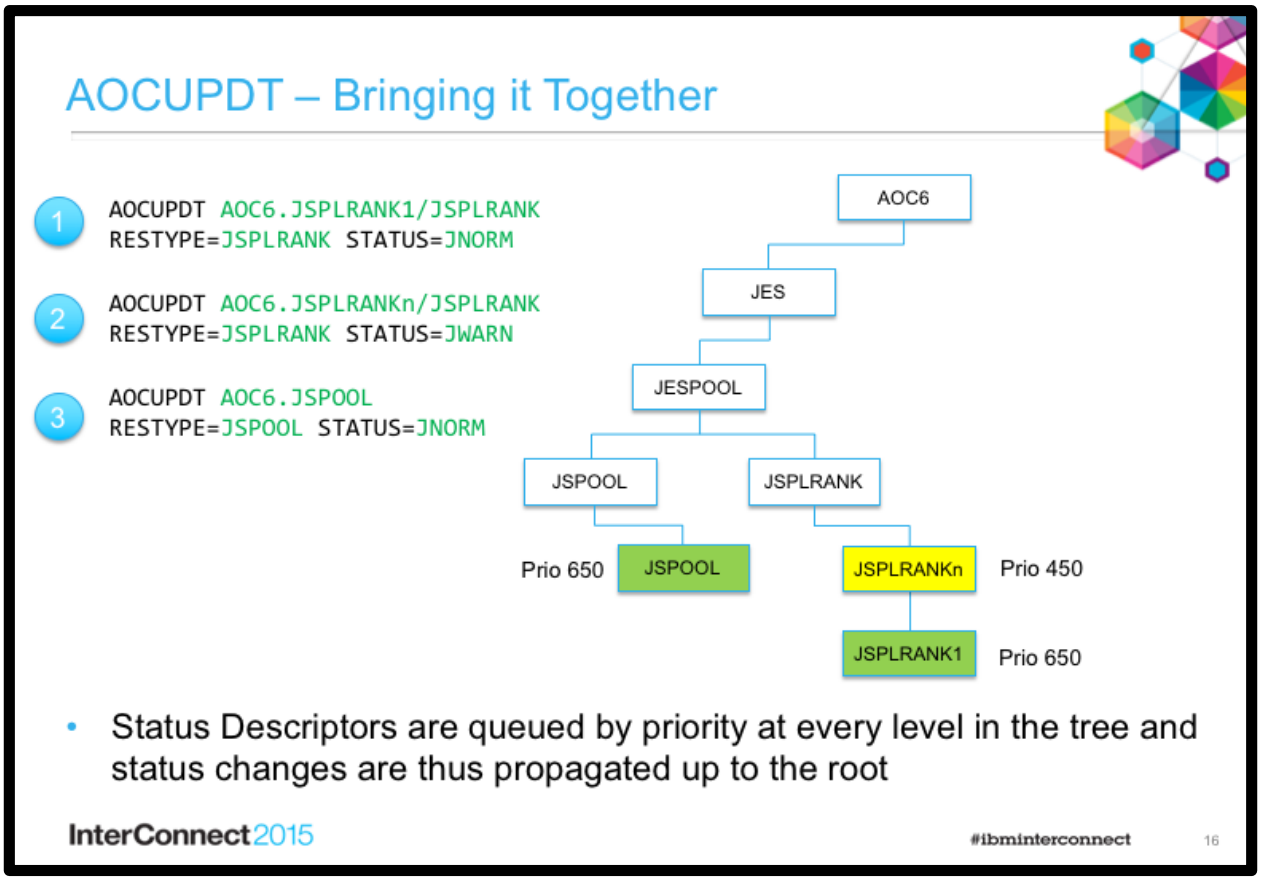


The appearance and the priority of the Status Descriptor is determined by the STATUS-parameter. SA z/OS already has a set of pre-defined status values that you can use also for your own purpose. It is unlikely that the pre-defined values are changed by the product. The values are documented in Priority and Color Default Assignments in the Programmer's Reference manual.

If you want to define your own status values, you can do so by defining a new SCR (Status Display) entry adding entries in your own SDF Details policy.

As illustrated by the screenshot, every status value has a name, a priority and a highlighting attribute. The Clear value indicates what AOCUPDT is supposed to do, whether existing Status Descriptors are replaced or not. It is possible to only replace descriptors that match exactly the new descriptor's reference value `(Y,RV)`. Alternatively, you can also replace all descriptor's that have the same prefix than the new descriptor's reference values `(Y,RV*)`.

To delete a Status Descriptor, you specify Clear Y and use Req NOADD. For further details on Clear and Req, have a look at this dialog's online help.



On this page, the status propagation is explained. Initially, no Status Descriptor exists, so all Status Components have a default (empty) status.

When the first AOCUPDT-call is made, a new component JSPLRANK1 is added into the tree with root AOC6. Since JSPLRANK1 itself is not defined in the tree, its superordinate component JSPLRANK is specified as the alternate component. JSPLRANK was chosen as the component reflecting the top-15 users of JES SPOOL space. A status of JNORM (as defined in the SCR-entry on the previous page) is used which will display this status in green color with normal highlighting.

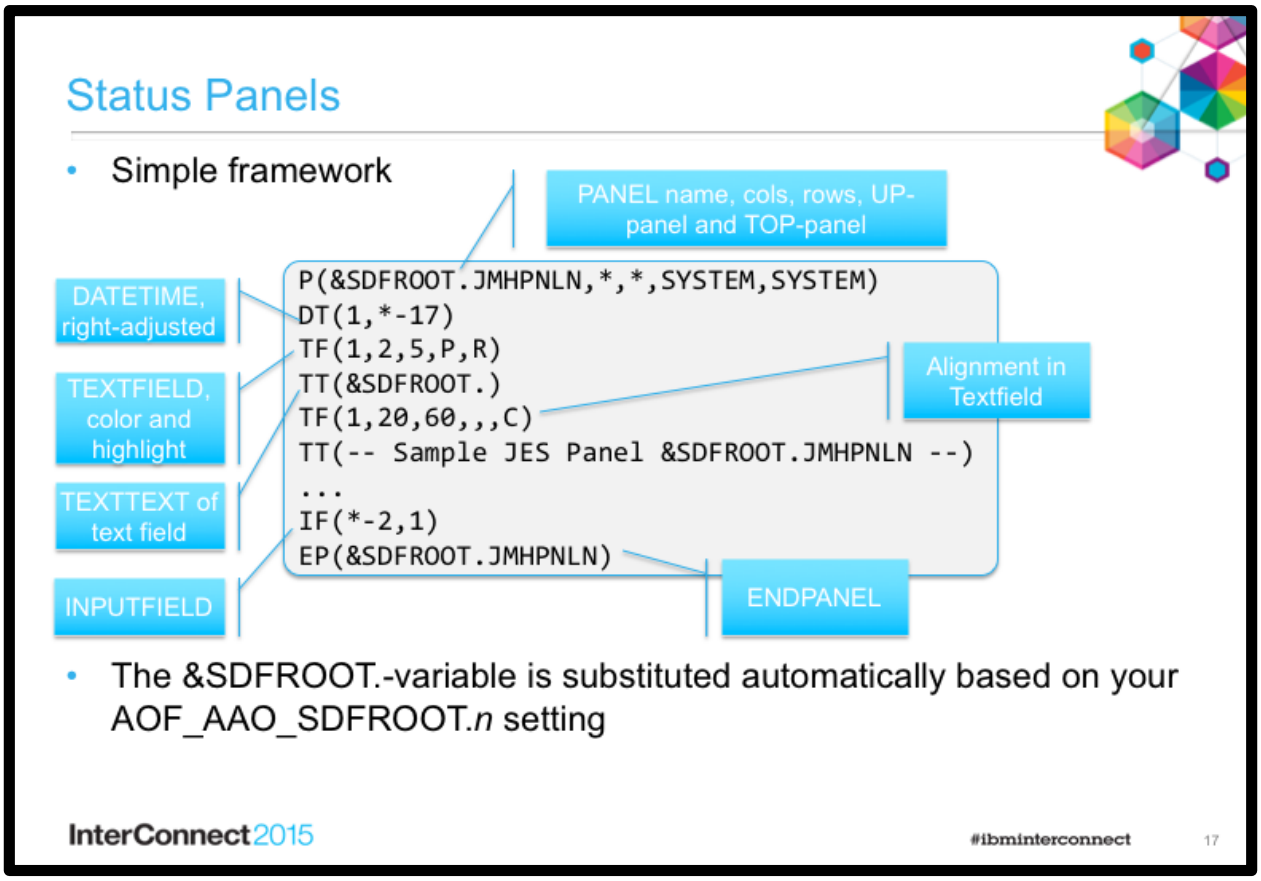
The Status Descriptor is not only chained to JSPLRANK, but also to all other Status Components in the propagation order specified in the AOFINIT DSIPARM-member. By default, propagation is done towards the root. This causes also to flip the color of the JESPOOL, JES and AOC6 components.

Then, another call of AOCUPDT is made. This time, the component JSPLRANKn is added into the same tree. Its status, however is JWARN. JWARN has a yellow color and a higher priority (450 instead of 650) than the already existing descriptor JSPLRANK1. Therefore it is chained ahead of JSPLRANK1.

Now, the highest status of JSPLRANK is that of JSPLRANKn, so its color changes to yellow. Because of the status propagation up the tree, the same happens for JESPOOL, JES and AOC6. An operator watching only an overview panel sees

immediately that the system is in a “yellow” state.

When the third call is made, the component JSPOOL is added into the tree. Its Status Descriptor will also be chained to all superordinate components, but since JNORM has a lower priority than the highest Status Descriptor that is already chained to JESPOOL, JES and AOC6, the status of the superordinate components doesn’t change. It remains JWARN (yellow).



So far, the structure of Status Components and how to update their status has been discussed. Next, we want to display the status to the operators.

SDF allows you to design your own panels. A typical template is shown on this page. It is a dynamic panel that you can recognize by the existence of the &SDFROOT. variable. During initialization time, a unique panel is created once for every root defined to the common global stem variable AOF_AAO_SDFROOT..

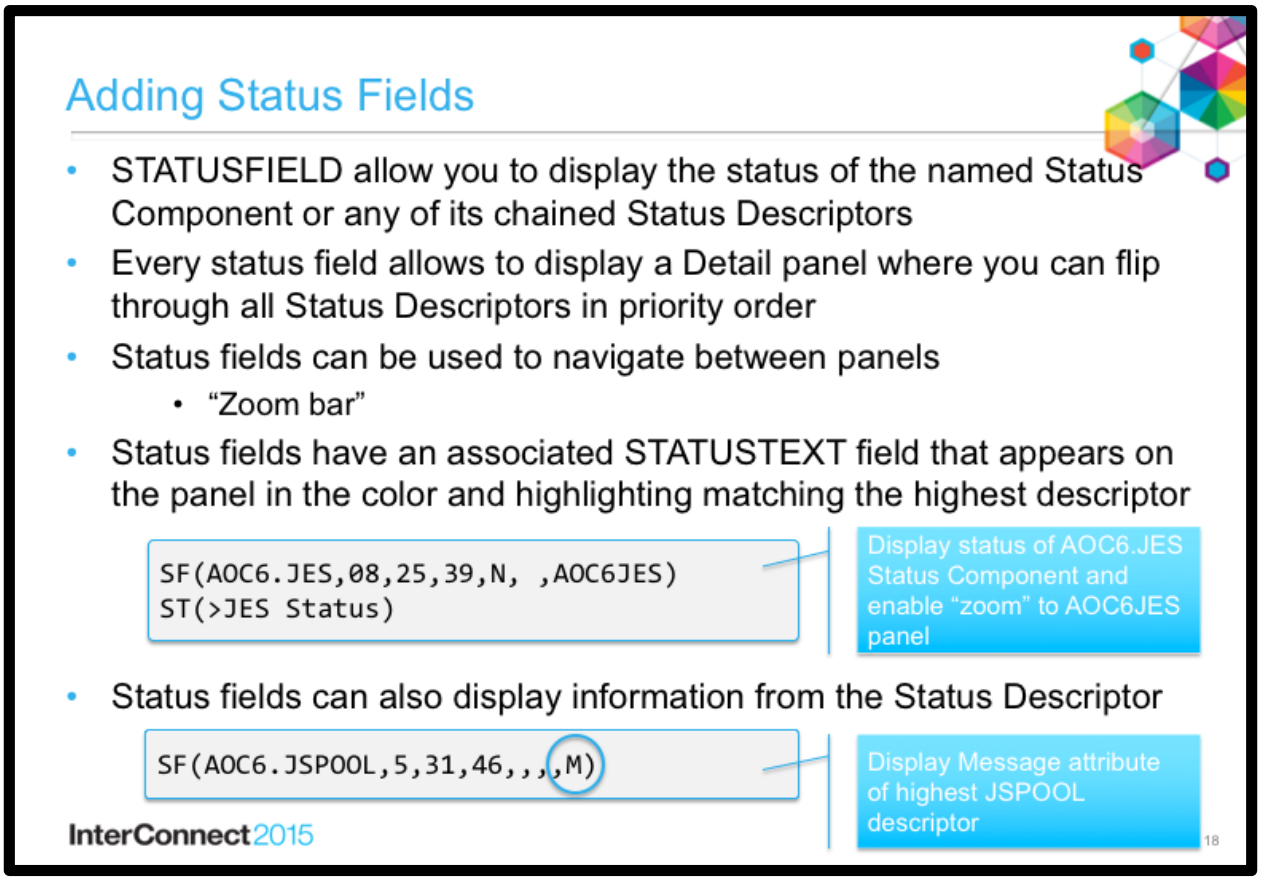
Every panel starts with a PANEL-statement and ends with an ENDPANEL-statement. The screen dimensions are not specified, so this panel adopts to the size of the actual terminal being used. Alternatively, you could specify fixed valus 24,80 for a 3278-mod 2 screen or 27,132 for a 3278-mod 5 screen. This panel points to the SYSTEM-panel for both, the UP-panel and the TOP-panel. So, using PF7 (UP) or PF12 (TOP, outside of a BODY) will return you to the SYSTEM-panel.

Often, you find date and time on the first line, right adjusted. You can define it using the DATETIME-statement. For operator orientation, it is good practice to also show the system name (here value of &SDFROOT.) on the title line. It is shown in columns 2 to 5 as static text. The location and dimension is specified using a TEXTFIELD-statement while a TEXTTEXT-statement is used to specify the actual literal being displayed.

The header is a piece of text including the name of the panel. It is centered within

columns 20 to 60. The alignment of the text can be specified in the TEXTFIELD-statement using ‘C’ as the last parameter.

Finally, every panel has an input field to allow operators enter commands. It is specified by the INPUTFIELD-statement. The input field is positioned on the second to last row using relative positioning.



To display the aggregated status of Status Component, you can use a STATUSFIELD-statement. This statement has a syntax like this:

```
SF(root.component,row,start_column,end_column,highlighting,
up_panel_name, down_panel_name, status_descriptor_number)
```

Some of the parameters have defaults, some are optional. The full parameter description can be found in the product manual.

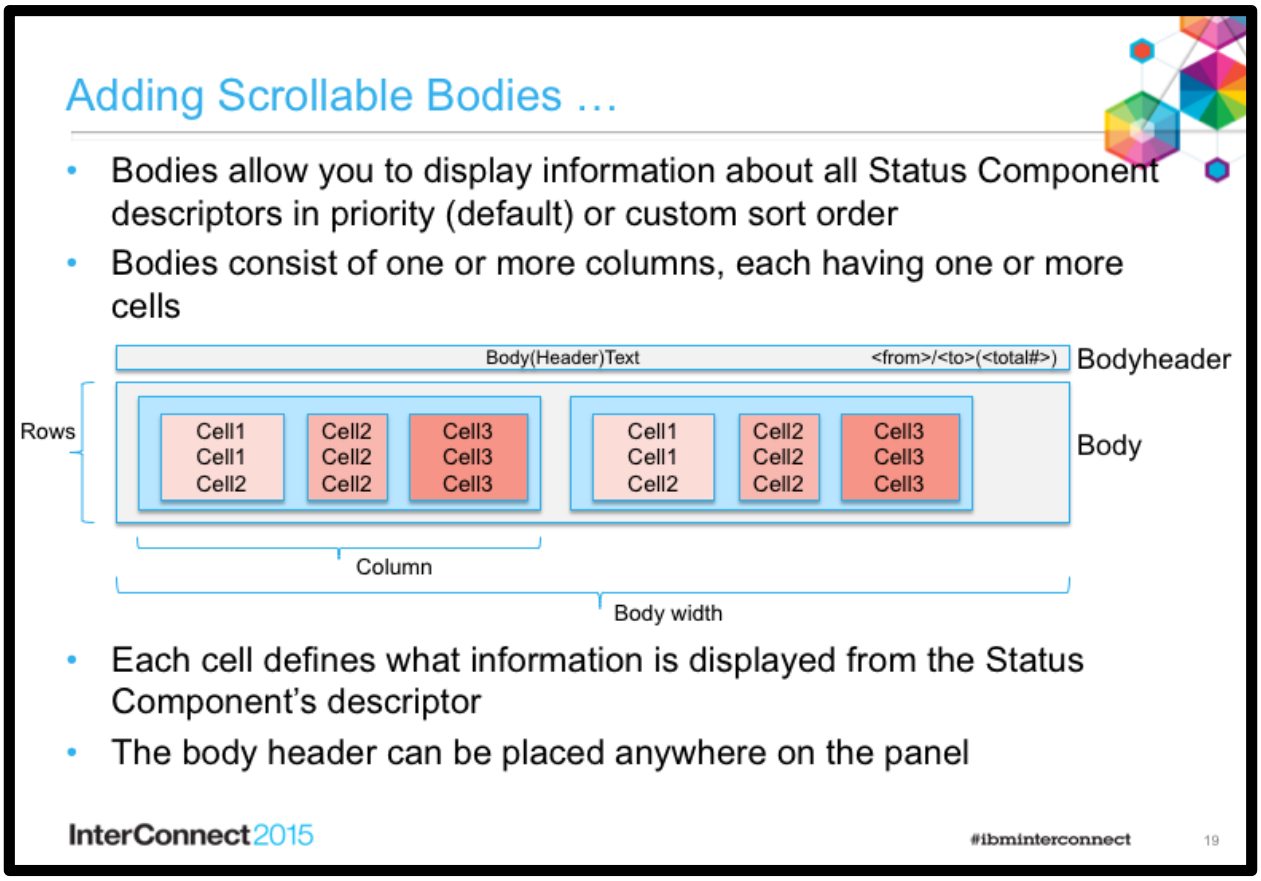
In the first example above, the aggregated status of the Status Component JES is shown in the Status Tree with root AOC6, representing system AOC6. The text on the panel is just ">JES Status" as told by the STATUSTEXT-statement. However, the color of this text depends on the actual highest Status Descriptor that is chained to the JES component. You can also see that the UP-panel name has been omitted and a DOWN-panel name has been specified, here panel AOC6JES. So, when the user tabs to this field and presses the PF8 ("Down") key, the panel AOC6JES will be invoked that can then show more details. If the user presses the PF2 ("Details") key, the Detail panel with the highest Status Descriptor will be shown. From the Info and Message fields you can immediately understand what the problem is (if any). You can also flip through all Status Descriptors chained under the JES component using PF8 ("Down") or PF7 ("Up"), jump to the last using PF11 ("Bottom") or back to the first using PF12 ("Top").

The second example shows how to display Status Descriptor attributes on the panel, here the message attribute from the highest Status Descriptor chained to Status Component JSPOOL in the same tree with root AOC6. The message attribute is denoted by the status descriptor number “M”. This is a shorthand notation for “0M”. In this case, the number is not used, as there is at most one single Status Descriptor chained to the JSPOOL-component. But using the ranked SPOOL users, for example, it would be conceivable to just show the utilization of the 2nd or 3rd user, represented by the respective Status Descriptor chained to the JSPLRANK-component. In this case, the STATUSFIELD would look like this, for example:

```
SF(AOC6.JSPLRANK,row,start,end,,,,2V)
```

```
SF(AOC6.JSPLRANK,row+1,start,end,,,,3V)
```

The Info attribute is displayed by default, which is why the optional “V” could be omitted. But specifying it is certainly better style.

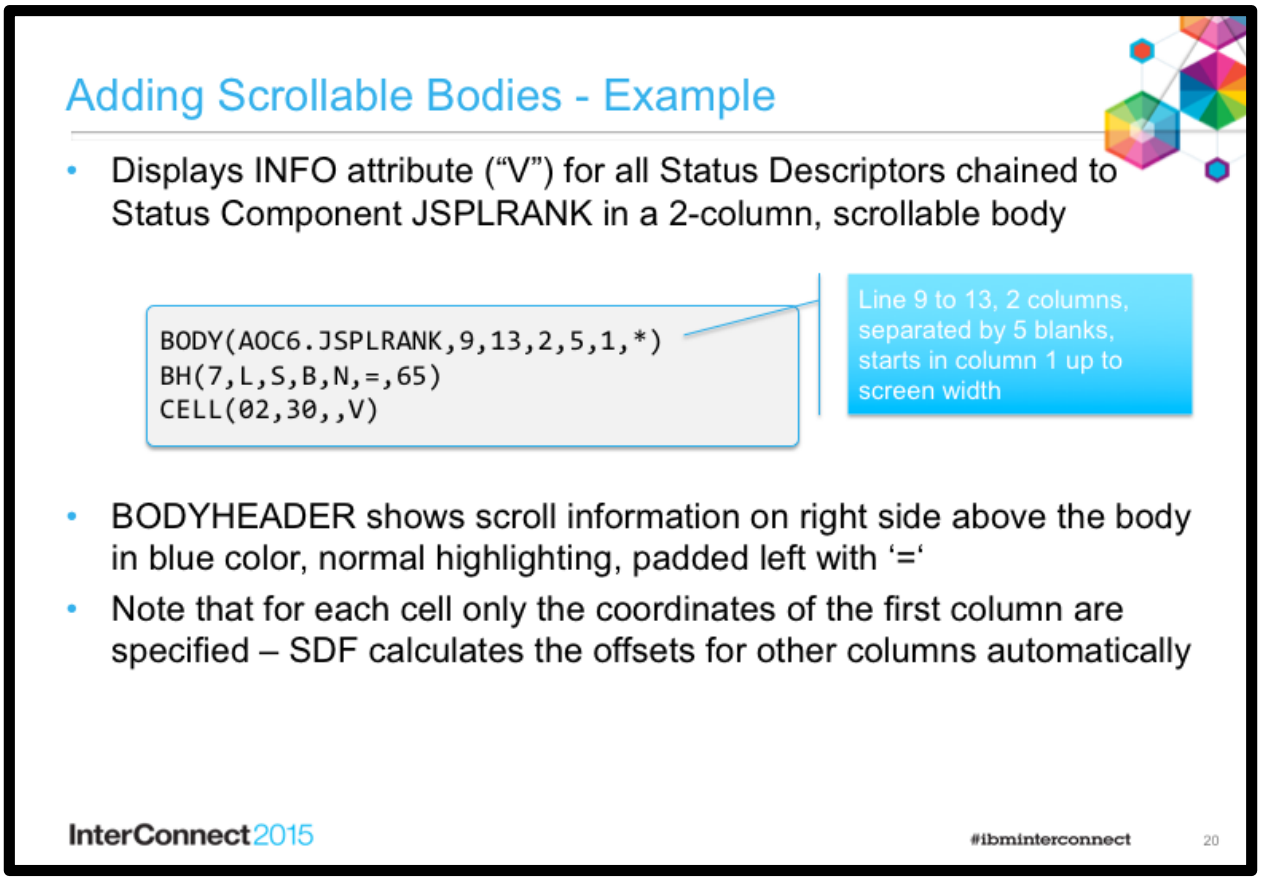


Defining STATUSFIELDs can be cumbersome, if you want or need to display many Status Components, for example one for every SPOOL user with a utilization greater than 1%. First, you don't know in advance how many these are and second, given the limited screen size, you will soon reach the limits of a panel. To be prepared for all cases, you'd have to specify as many STATUSFIELDs as you need and use multiple panels that you can connect using LEFT and RIGHT panel names in the PANEL-statement.

Since a while, SDF provides an alternative for such use cases. The BODY-statement can be used to specify the location and dimension of a scrollable area. In this area, you can display any Status Descriptor attributes for a given Status Component.

In its simpliest form, the BODY consists of a single column which in turn is divided into one or more cells. Each cell displays one single Status Descriptor attribute. If you have enough room on the screen, you can specify how many columns this body should have and SDF replicates the layout of the first column to the right as long as the column fits completely on the actual screen. Status Descriptor information is then shown from top to bottom and from left to right. When the cursor is placed in the BODY-area, and you press PF9 ("Right") the next set of Status Descriptor information is shown and when you press PF10 ("Left") the previous set is shown. PF11 gets you the rightmost set of information and PF12 to the leftmost set of information.

Each body can have a body header, denoted by a BODYHEADER-statement. It is optional, but of course useful, because you can place header text in it (using BODYTEXT), visually separate this body from another on the same panel, using dashes for example, or just because it is very convenient to see scroll information. The body header can be located anywere on the panel, it doesn't have to be defined right above its body.



Here is an example of a body displaying the top-n list of JES SPOOL users. The BODY-statement syntax is like this:

```
BODY(root.component,first_row,last_row,num_columns,column_distance,start,end)
```

The fields `start` and `end` denote the start and end screen columns of the body. The asterisk instructs SDF to basically expand the body to the right up the limits of the screen. On a 24x80 screen, the body reaches out to column 80 while on a 27x132 screen, the body reaches out to column 132.

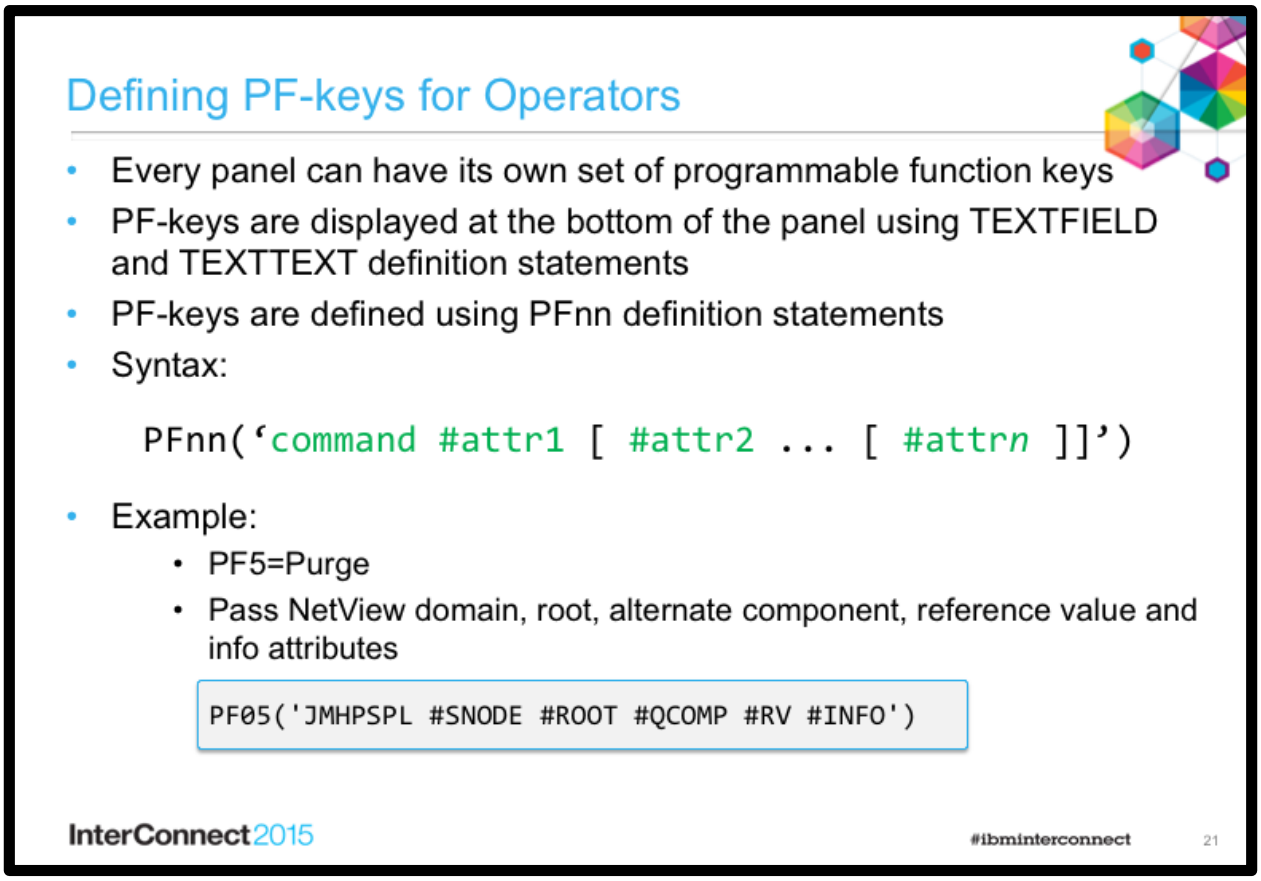
The body above will show two columns and the distance between these columns is 5 characters. Within each column, there is a single cell that starts on the 2nd character in the column and ends on the 30th. The Status Descriptor attribute shown in this cell is the Info attribute ("V").

Finally, the BODYHEADER-statement causes a header line to be shown in row 7. The syntax is:

```
BODYHEADER(row,alignment,scroll_info,color,highlight,padding_characters,start)
```

In the example above, the header starts on column 65, shows scroll information and is padded with "=" on the left. Scroll information is always right adjusted. The full

parameter description for BODY and BODYHEADER can be found in the product manual.



No console is useful for operators, if they cannot also interact with it if need may be. SDF allows you to define your own set of PF-keys to issue commands in the context of the Status Component, where the cursor is currently located at. Like any other static text, the list of PF-keys valid for a panel can be displayed as usual TEXTFIELDs at the bottom of the panel, just below the input field.

You can define up to 24 PF-keys. Basically, the definition consists of one single string. The first word in the string is the command. Everything else are command parameters. The parameters can be a mix of literals or variables. SDF provides a variable for every Status Descriptor attribute. These variables are described in the product manual where the PFKnn-statement is documented.

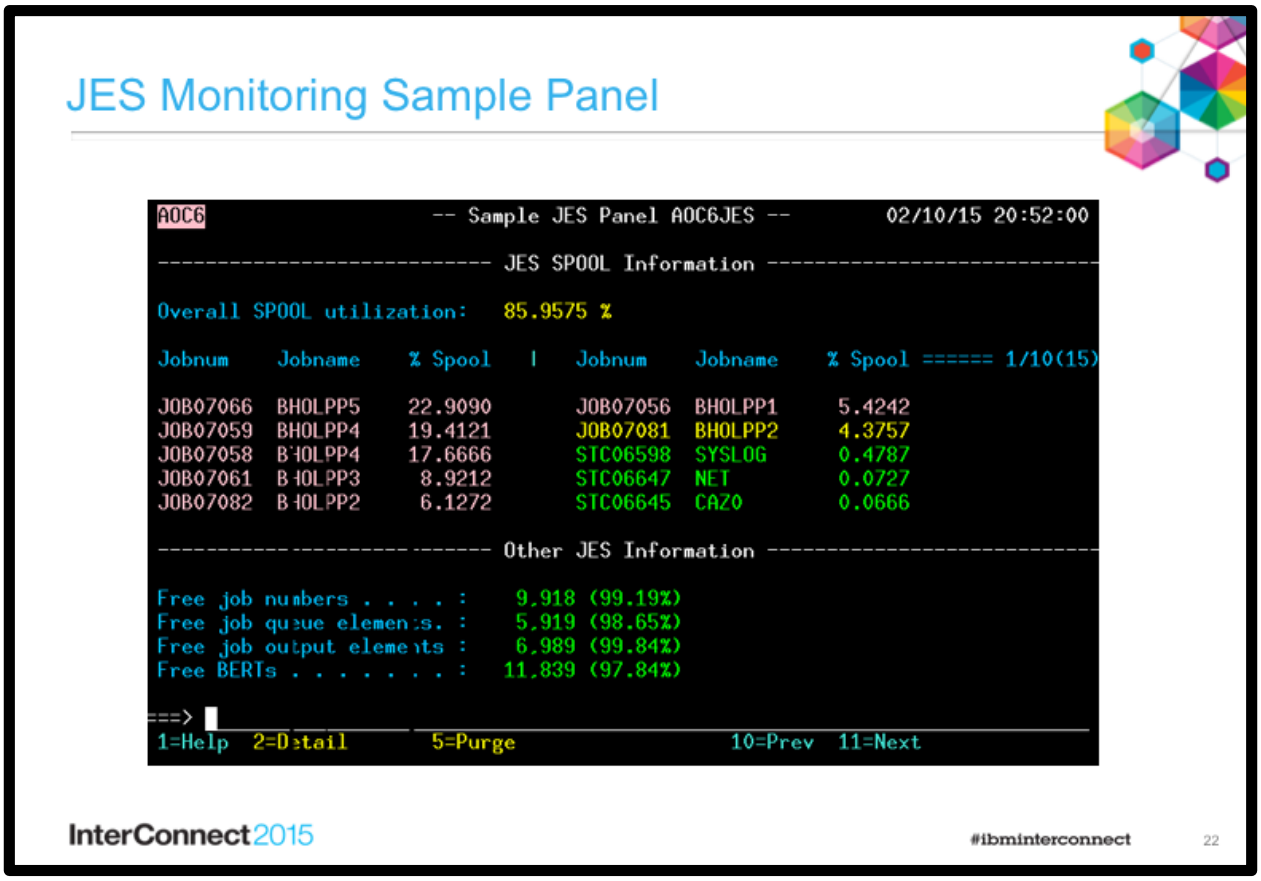
The variables can be specified either with an ampersand (&) or with a hash sign (#) as used in the example above. Use of hash is recommended to not confuse them with possible system symbols.

Note, the command is always executed on the system, where SDF is running, which can be the focal point system. So, when you want to purge the SPOOL output of a job on AOC6, while being logged on to AOC7, for instance, the actual JES-purge command must be sent to AOC6 and not executed locally. Therefore parameters like the NetView domain (#SNODE) need to be passed to JMHPSPL. The reference value, containing the job number, is passed with #RV. These two parameters are sufficient to send the

```
MVS $PO <jobnum>
```

command to the system, where this job consumes space on the SPOOL data set. The other parameters #ROOT, #QCOMP and #INFO are passed such that the JMHPSPL can also delete the Status Descriptor from the display. Remember, AOCUPDT requires that you specify the tree and the Status Component to which a new Status Descriptor should be added. The pre-defined status DELETE ensures that existing Status Descriptors with this reference value are deleted. Here is an example, how JMHPSPL would get rid of the Status Descriptor for component JSPLRANK1:

```
AOCUPDT AOC6.JSPLRANK1/JSPLRANK STATUS=DELETE RESTYPE=JSPLRANK
RV=JSPLRANK1
```



The panel above demonstrates the capabilities of SDF at the example of a simple JES status monitor. Here is the complete panel definition:

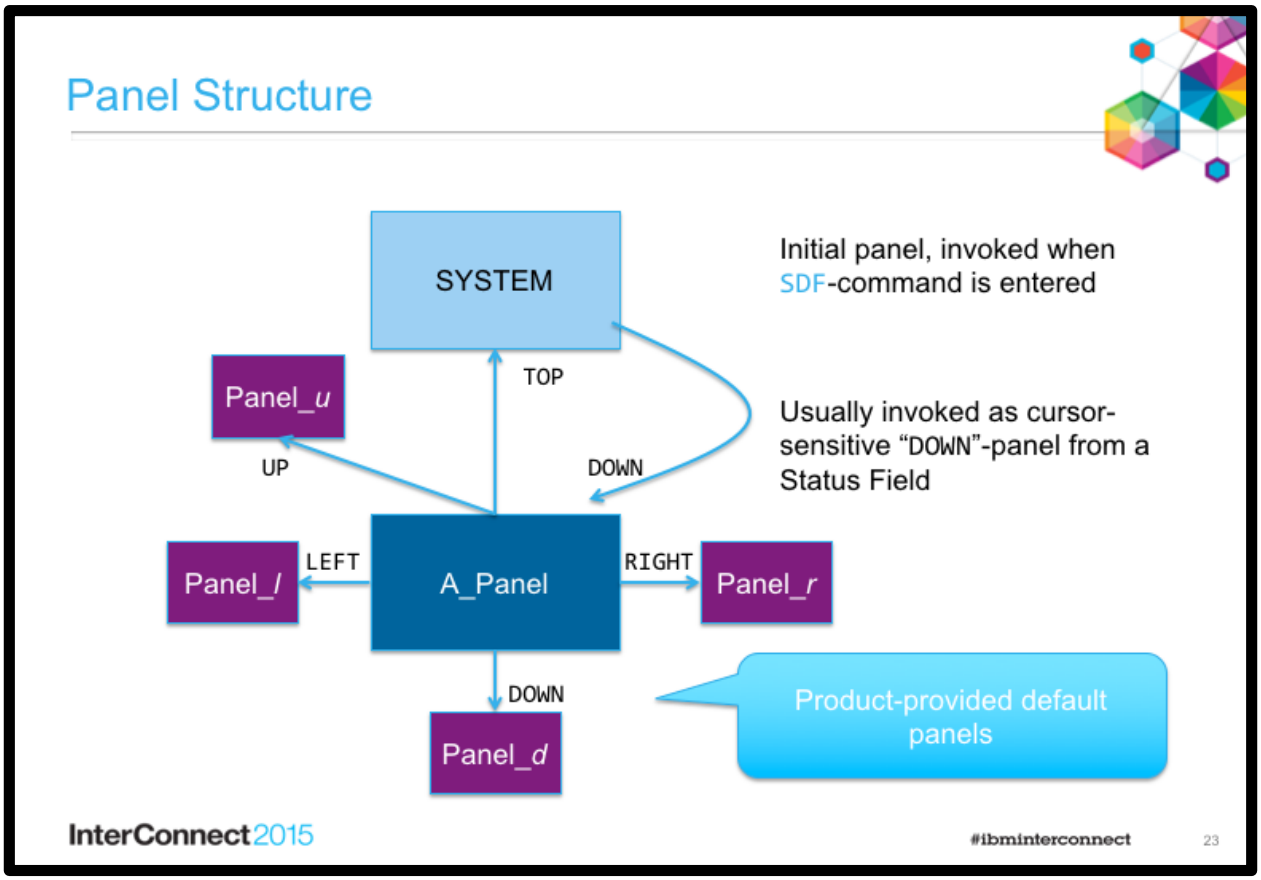
```
P(&SDFROOT.JES,24,80,SYSTEM,SYSTEM)
DT(1,*-17)
TF(1,2,5,P,R)
TT(&SDFROOT.)
TF(1,20,60,,,C)
TT(-- Sample JES Panel &SDFROOT.JES --)
/* ------------------------------------------------------------
----- */
TF(3,2,80,N,N,C,-)
TT(JES SPOOL Information)
TF(5,2,29,B,N)
TT(Overall SPOOL utilization: )
SF(&SDFROOT..JSPOOL,5,31,46,,,,M)
TF(7,2,30,B,N)
TT(Jobnum    Jobname    % Spool)
TF(7,33,33,T,N)
TT(|)
TF(7,37,64,B,N)
```

```
TT(Jobnum    Jobname    % Spool)
BODY(&SDFROOT..JSPLRANK,9,13,2,5,1,*)
BH(7,L,S,B,N,=,65)
CELL(02,30,,V)
/* ------------------------------------------------------------
----- */
TF(15,2,80,N,N,C,-)
TT(Other JES Information)
TF(17,2,27,B,N)
TT(Free job numbers . . . . :)
SF(&SDFROOT..JSJNUM,17,29,45,,,,V)
/* ------------------------------------------------------------
----- */
TF(18,2,27,B,N)
TT(Free job queue elements. :)
SF(&SDFROOT..JSJQE,18,29,45,,,,V)
/* ------------------------------------------------------------
----- */
TF(19,2,27,B,N) TT(Free job output elements :)
SF(&SDFROOT..JSJOE,19,29,45,,,,V)
/* ------------------------------------------------------------
----- */
TF(20,2,27,B,N)
TT(Free BERTs . . . . . :)
SF(&SDFROOT..JSBERT,20,29,45,,,,V)
/* ------------------------------------------------------------
----- */
PF05('JMHPSPL #SNODE #ROOT #QCOMP #RV #INFO')
IF(*-2,1)
TF(*-1,1,8,T,N)
TT( 1=Help)
TF(*-1,50,79,T,N)
TT(10=Prev  11=Next)
TF(*-1,10,17,Y,N)
TT(2=Detail)
TF(*-1,25,32,Y,N)
TT(5=Purge)
EP(&SDFROOT.JES)
```

This panel can be added to your main system panel using a status field similar to the example shown on page “Adding Status Fields”.

The sample script JMHPSPL and also the script that collects the JES status shown on

the panel are provided as collateral to this presentation.



SDF can consist of as many panels as you want and need. If defined accordingly, every panel allows to navigate to its left and right neighbor and also to its parent and (first) child. When the operator enters the SDF-command, the initial panel will be displayed. By default, it is called SYSTEM. If you don't like the name, you can change it but you have to revisit your AOFINIT-parameter member as well and change the INITSCRN-setting, accordingly.

To quickly navigate to the initial panel from anywhere, that panel can be defined as the "TOP" panel in every other panel.

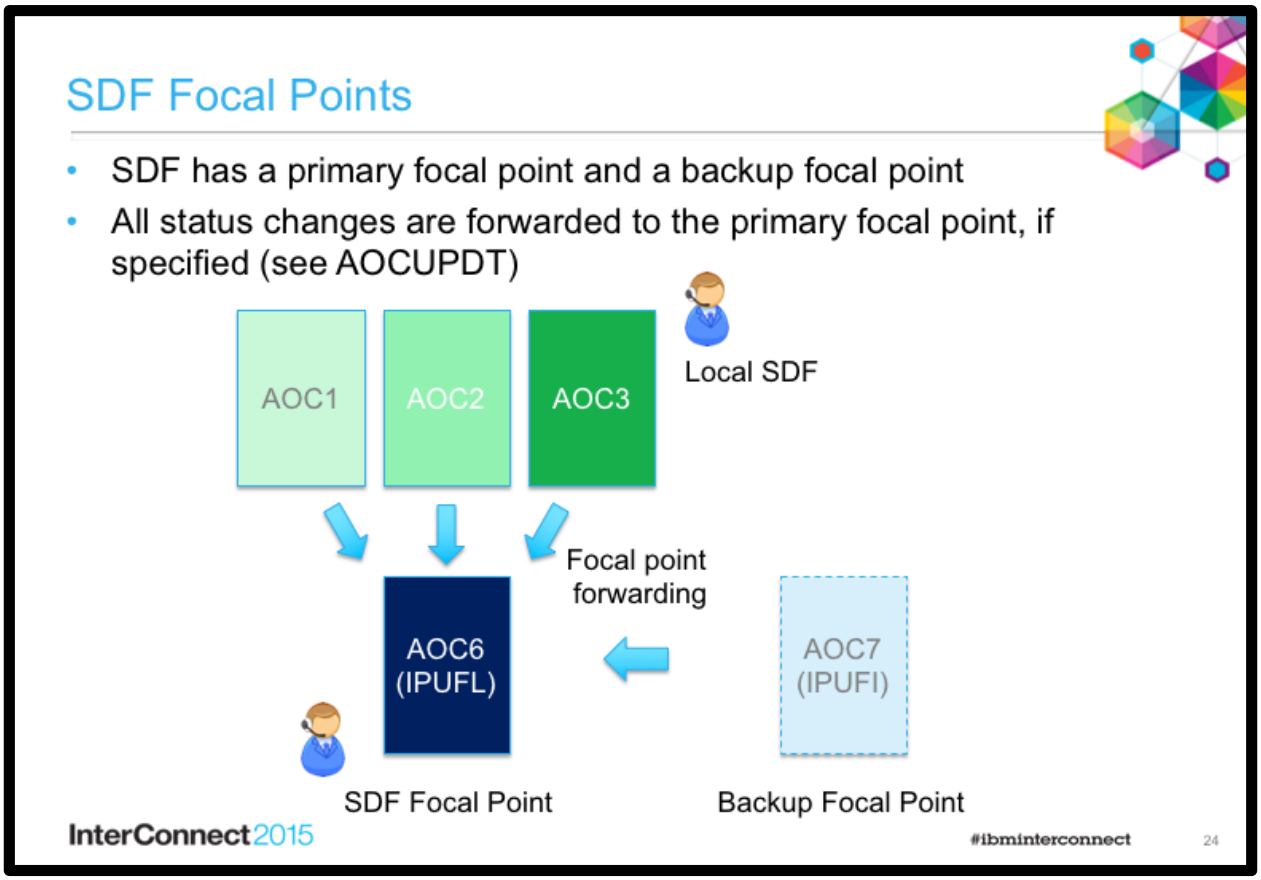
By default, the PF-keys are defined like this:

- PF7 = UP
- PF8 = DOWN (also called "Zoom")
- PF10 = LEFT (also called "Previous")
- PF11 = RIGHT (also called "Next")
- PF12 = TOP

Note, some PF-keys have different meanings depending on the context being used. The behaviour can differ whether the cursor is located on a status field or within a body or elsewhere on the panel. Pressing PF1 ("Help") will explain how the PF-keys work.

The ability to define “LEFT” and “RIGHT” panels has historical reasons, when scrolling wasn’t supported by SDF. Administrators used this technique to chain multiple panels in sequence to display as many resources and related information as needed by the installation. Given that the screens used to be only 24 lines by 80 columns, quite a number of panels were needed.

Today, you can define panel bodies that support scrolling (up, down and left, right) when the cursor is placed in the body area, without having to define additional panels, as you have seen before.



The focal point concept was already mentioned several times before, but what is it exactly? In the SA z/OS customization dialog, you define a primary and backup system that both can act as the *SDF focal point*. To define an SDF focal point, you create a new or modify an existing type NTW (network) entry and fill out the SDF FOCALPOINT policy. Here you define the primary and backup NetView domains. For the illustration above, the following would be defined:

```
Primary domain = IPUFL
```

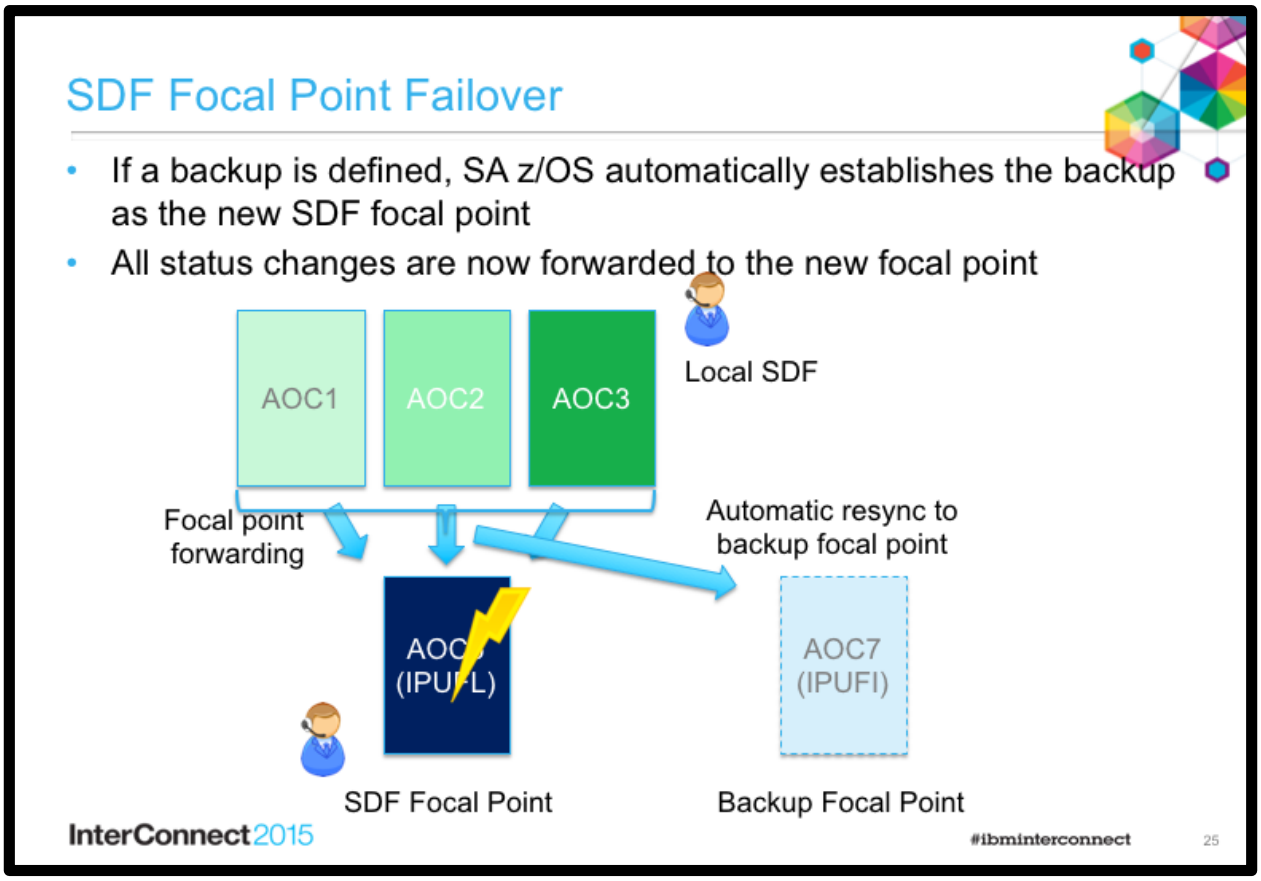
```
Backup domain = IPUFI
```

This NTW-entry must be linked to all systems in the environment, here to the systems AOC1, AOC2, AOC3, AOC6 and AOC7. Every system has a local SDF console but only SDF on system AOC6 provides the complete enterprise view.

Status updates that occur on any of the systems cause SDF updates locally. If focal point forwarding was specified with AOCUPDT, then the status update is also sent to the SDF focal point on AOC6 and the panels there are updated accordingly. Note that SA z/OS status updates are always forwarded to the focal point if it exists.

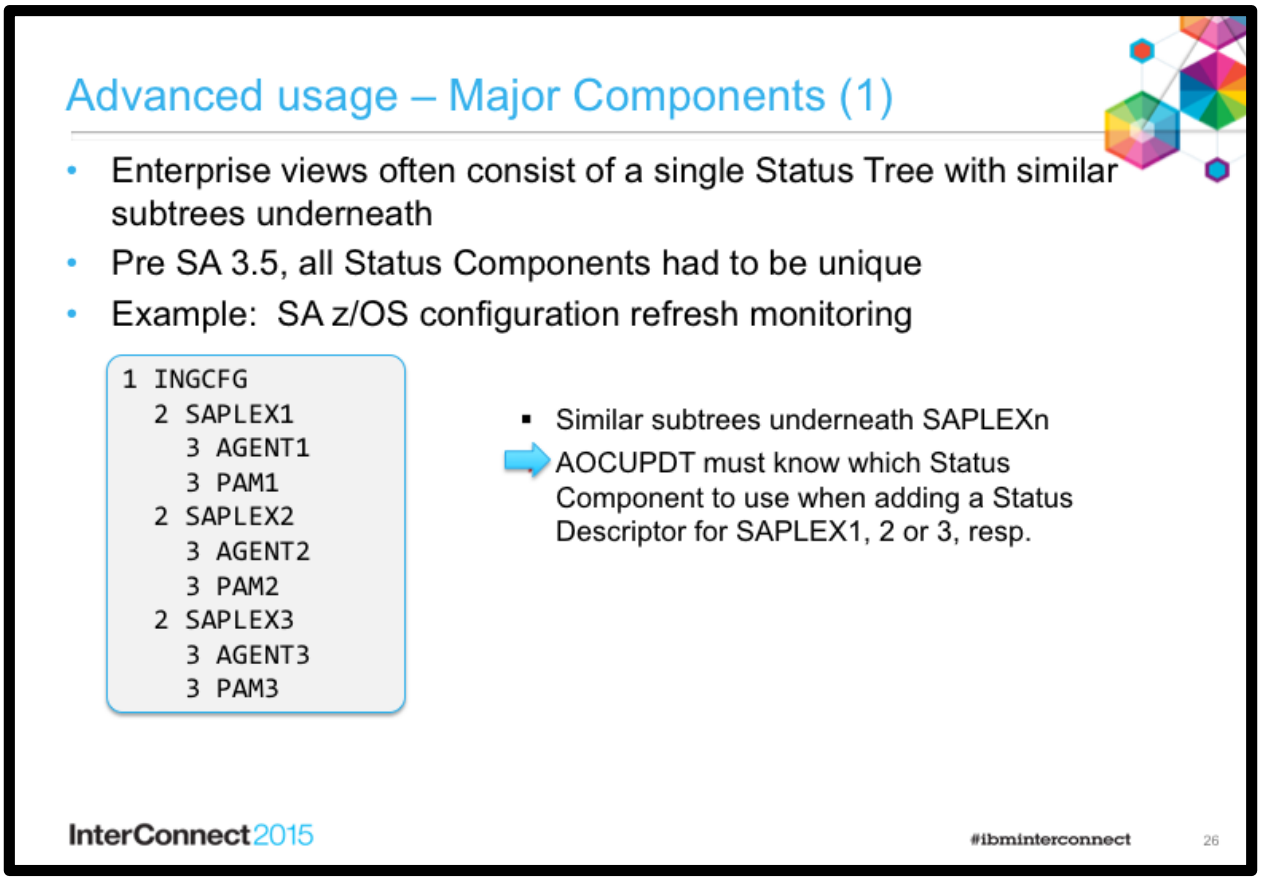
The difference between the focal point system and, for instance, AOC1 is that AOC1's SDF root list only specifies AOC1, while the SDF root list for AOC6 and also AOC7 specifies all 5 systems. Using dynamic panels, the very same system-related panels can be reused by all systems and need not be implemented multiple times. The same

is true for dynamic Status Trees.



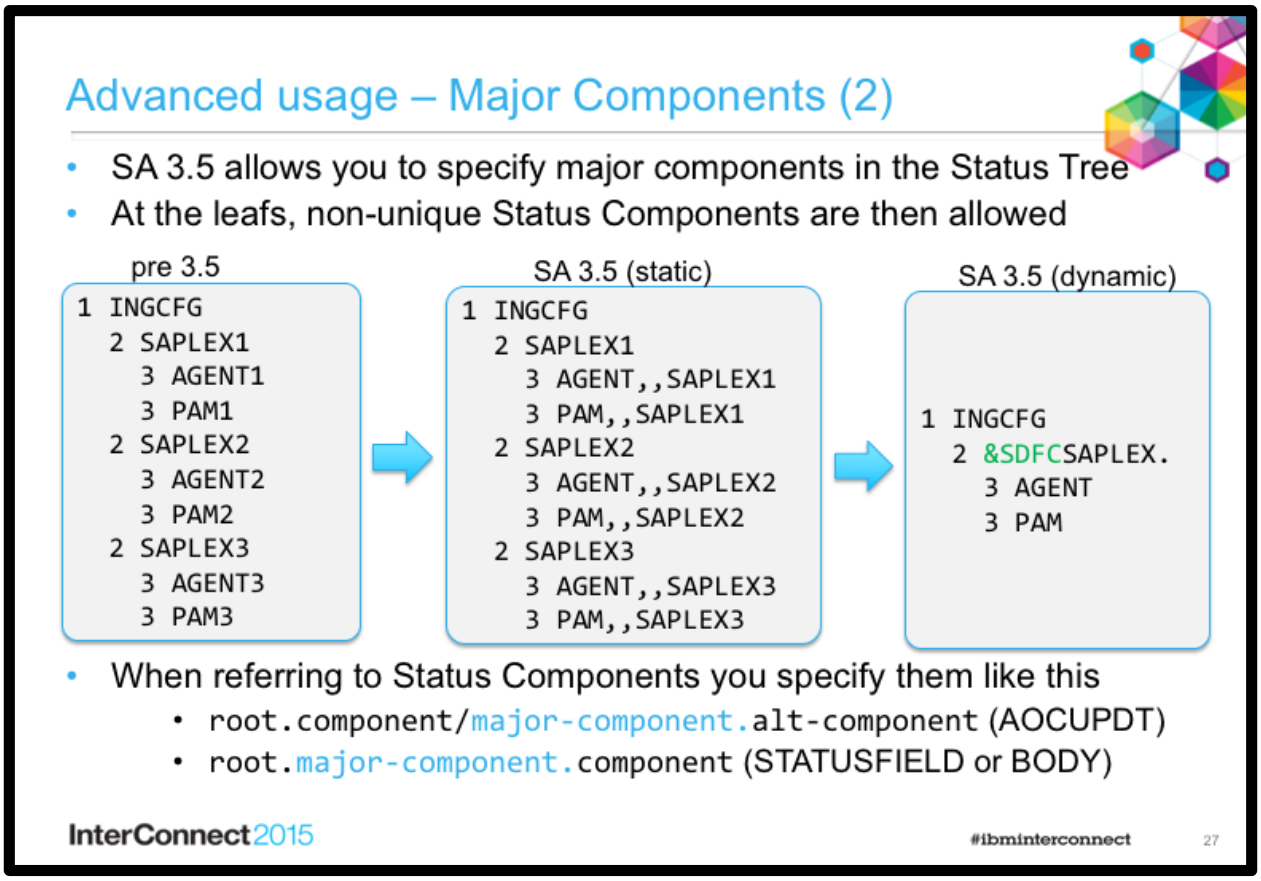
When the primary SDF focal point fails, the status is automatically moved over to the backup focal point. Internally, the RESYNC FP command is executed on all systems in order to synchronize the local status with the status on the new focal point system, in the example AOC7.

With SA z/OS V4.1.0 and OA55386, it is also possible to configure SDF such that both focal points, primary and backup, are updated in parallel. Once parallel update is turned on, no failover is needed and both focal points always present the same status. Refer to “Appendix B. Customizing the Status Display Facility” in the Customization and Programming manual for more information about activating this function and also for configuring the focal points in the right way such that they can mutually forward any Status Component update to each other depending on their role.



Before SA z/OS V3.5, all Status Components in the Status Tree had to be unique. This led to artificial names for leaf components, that are all structured in like subtrees. A typical example are subtrees representing SA subplexes with subplex-specific components underneath. The example illustrates automation agent configuration status (AGENT1, AGENT2 and AGENT3) and also primary automation manager status (PAM1, PAM2, PAM3).

It is more intuitive to allow a means where identical subtrees can be replicated similar to the replication of whole trees using the &SDFROOT. variable. Such a means has been introduced with SA z/OS V3.5 and is referred to as *major component*.



Major components can be specified statically as shown in the middle part of the picture above. But as you see, this doesn't necessarily reduce the complexity of the Status Tree comparing it with the tree that you would have created pre SA z/OS V3.5.

However, SA z/OS also introduced variables &SDFCxxx.n that you can use similar to &SDFROOT. You specify these variables in the stem AOF_AAO_SDFCmyvar.n, for example. During initialization, the subtree containing &SDFCMYVAR. will be replicated and the actual token from the stem variable will be inserted as the Status Component.

SA z/OS demonstrates the usage of major components with its configuration refresh monitoring function. If we had 3 SA subplexes, then the variable AOF_AAO_SDFCsaplex would be defined like this:

```
AOF_AAO_SDFCsaplex.1 = "SAPLEX1 SAPLEX2 SAPLEX3"
AOF_AAO_SDFCsaplex.0 = 1
```

Note that none of the names can be longer than 16 characters, which is the limitation for any Status Component name.

In the Status Tree, SA z/OS resolves all variables with the name &SDFCSAPLEX by the tokens in the AOF_AAO_SDFCsaplex. list.

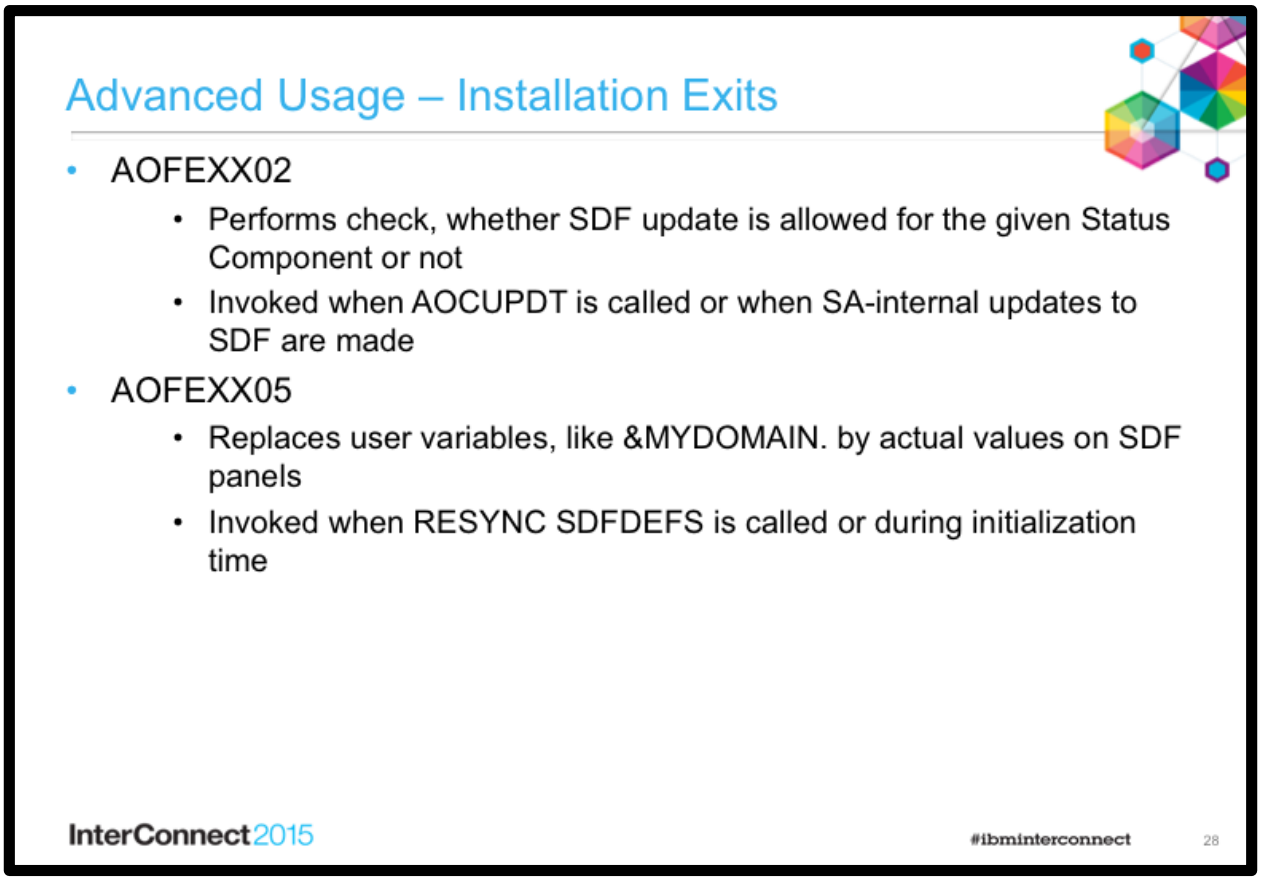
To update Status Components using AOCUPDT, the major component must now be

specified as follows:

```
root.component/major-component.alt-component
```

When referring to such Status Components in STATUSFIELDs or BODYs, you specify the major component as middle qualifier like this:

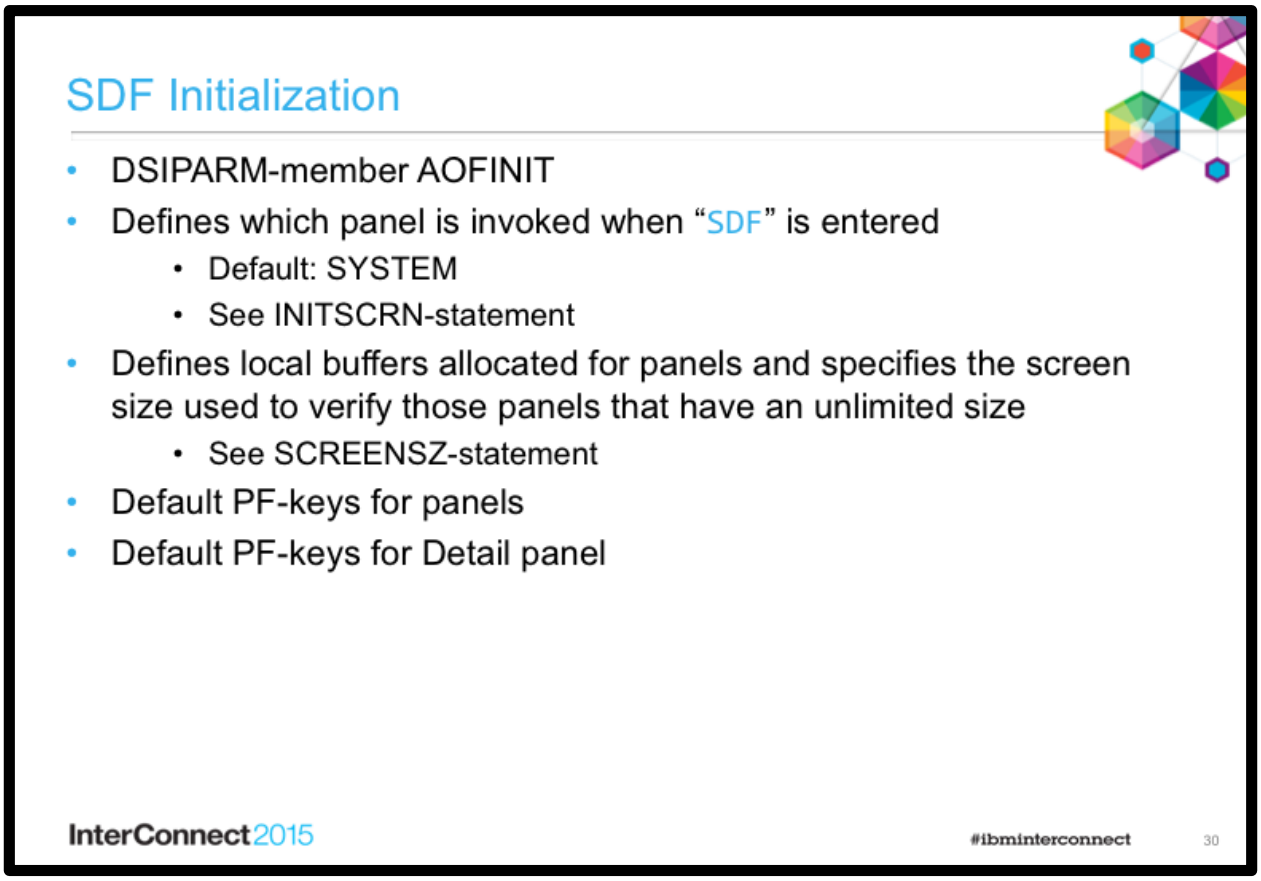
```
root.major-component.component
```



SA z/OS provides two installation exits that you can use to control what status is displayed (AOFEXX02) or to resolve user variables on panels or trees (AOFEXX05).

When implementing AOFEXX02, either recycle NetView or use the command RESYNC EXITS to load the exit. It is not sufficient to just place the exit routine into the DSICLD library concatenation.

The next section gives a brief overview on administering of SDF. For more details, please refer to the product manuals.

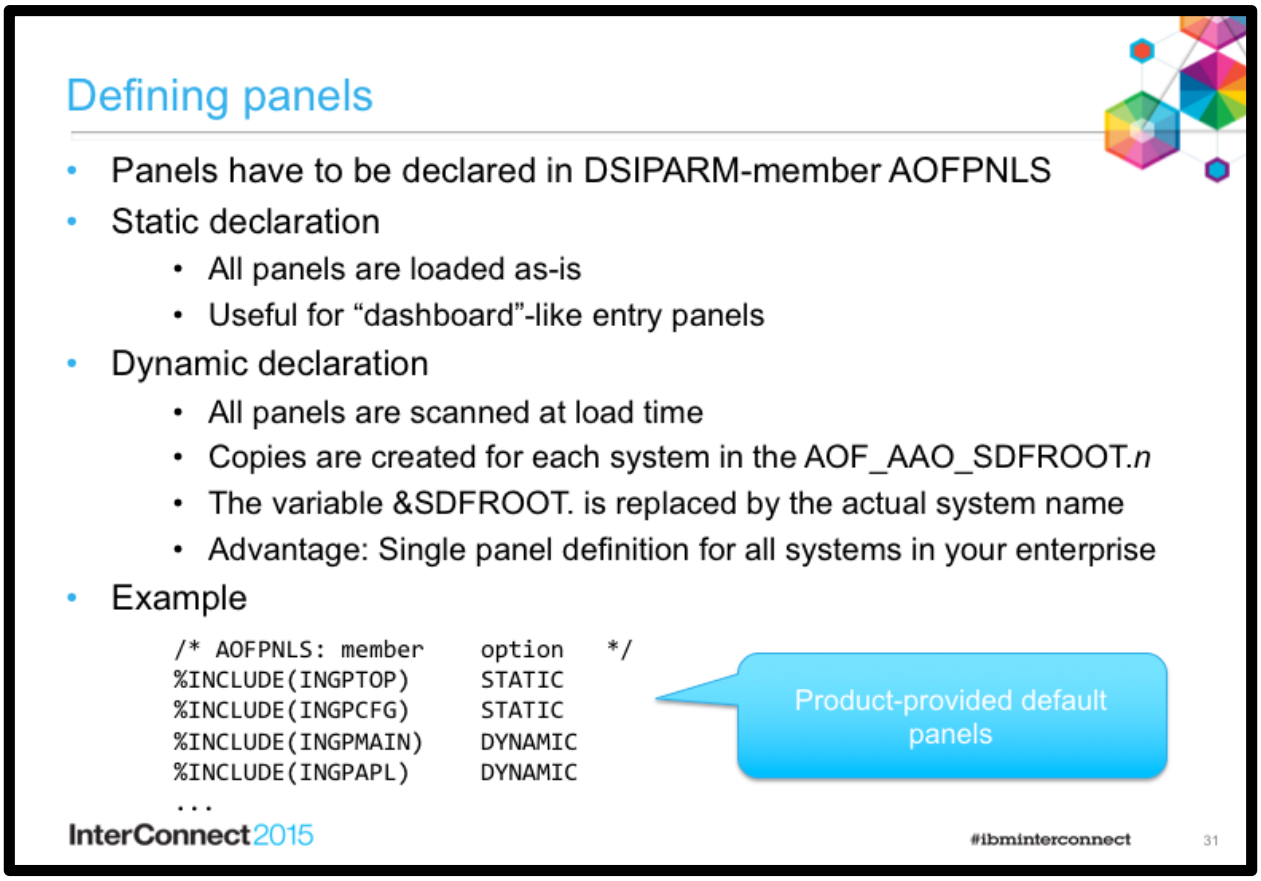


General SDF parameters are specified in the DSIPARM-member AOFINIT. When the operator enters SDF, the panel specified in the INITSCRN-statement is displayed. By default, the panel is called SYSTEM. See INGPTOP in SINGSAMP for the product-provided SYSTEM-panel.

The SCREENSZ-statement is used to specify the size of local screen buffer for a single SDF user. Unfortunately, there is no good method to guess the size. The default should be sufficient in most cases, however. The product manual describes a complex but exact method to calculate the buffer size.

There is normally no reason to change the other default settings, like priority ranges, direction for propagation or PF-key assignments. But AOFINIT is the place, where you can do this.

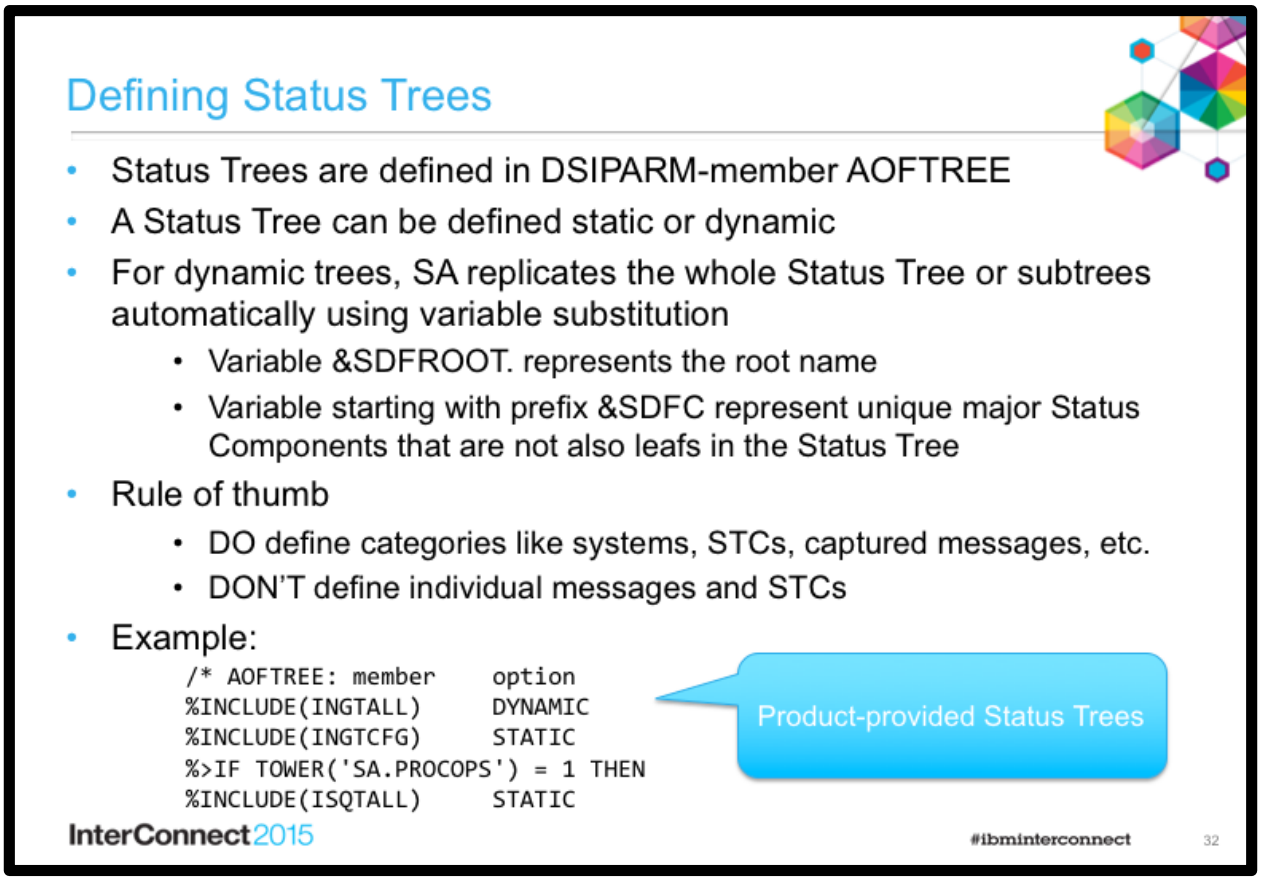
For more details on AOFINIT, refer to "Chapter 5. SDF Initialization Parameters" in the Programmer's Reference manual.



The DSIPARM-member AOFPNLS contains all the panels loaded by SDF. Use AOFPNLS as including member only, i.e. you include other panels rather than defining the panels in AOFPNLS themselves. Refer to the AOFPNLS-member in SINGSAMP for an example.

As mentioned before, there are static panels and dynamic panels. For the dynamic panels, the panel code is replicated and the &SDFROOT. variable is used in the panel name itself but also in all referenced panels to resolve to unique panel names.

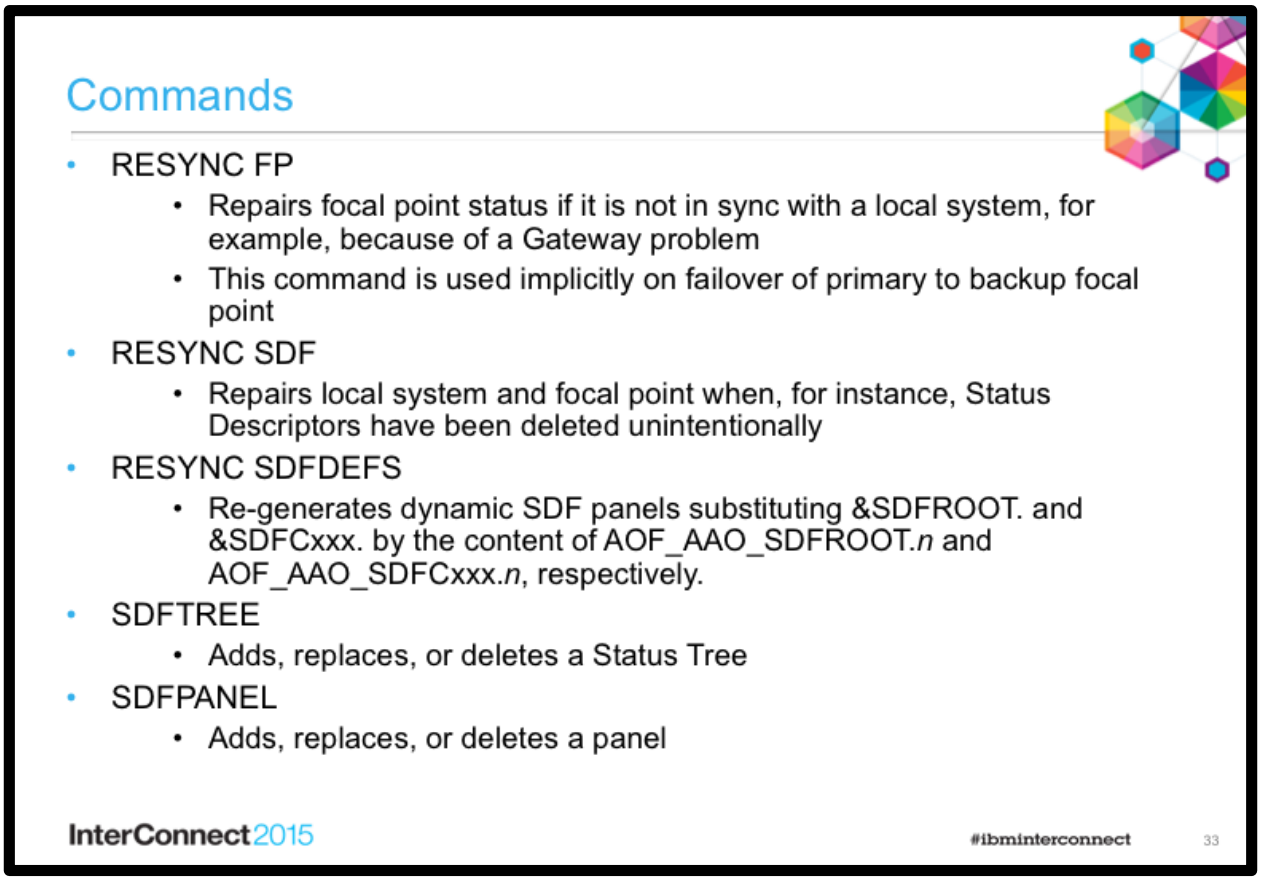
For more details on AOFPNLS, refer to "Appendix B. Customizing the Status Display Facility" in the Customization and Programming manual.



The DSIPARM-member AOFTREE contains the Status Trees used by SDF. Use AOFTREE as including member only, i.e. you include other trees rather than defininging the trees in AOFTREE themselves. Refer to the AOFTREE-member in SINGSAMP for an example.

As mentioned before, there are static trees and dynamic trees. For the dynamic trees, the whole tree is replicated and the &SDFROOT. variable is used as the level-1 root Status Component of the tree. Also when the variable &SDFCxxx. appears in a subtree, the subtree is replicated and the variable &SDFCxxx. is used as a unique major component in the status tree.

For more details on AOFTREE, refer to “Appendix B. Customizing the Status Display Facility” in the Customization and Programming manual.



This page gives an overview of the most important commands.

RESYNC FP

Use this command, when the local status doesn't match the focal point's status but the local status is OK. Local status is simply forwarded to the focal point.

RESYNC SDF

Use this command, when the local status doesn't match the focal point's status or the local status is not OK. Complete status monitoring is performed locally and all status is then forwarded to the focal point.

RESYNC SDFDEFS

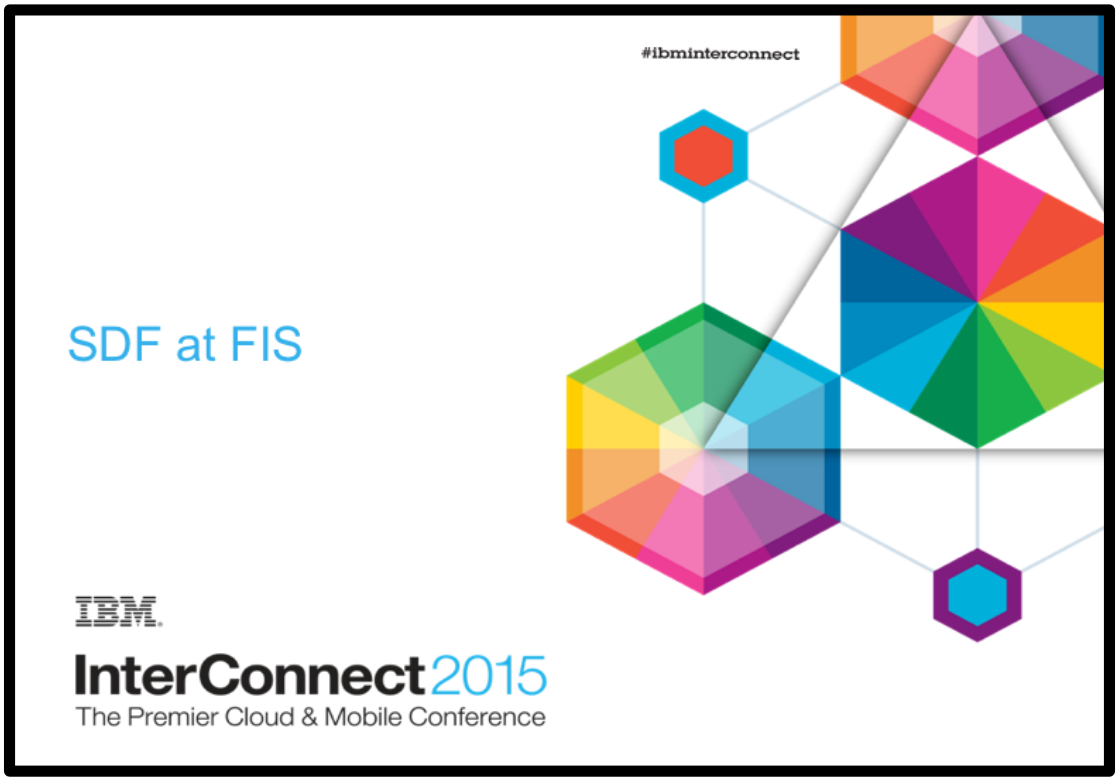
Use this command, when you have changed AOF_AAO_SDFROOT., AOF_AAO_SDFCxxx., panels or trees and want to re-drive dynamic panel and tree processing. All panels and trees are re-created anew.

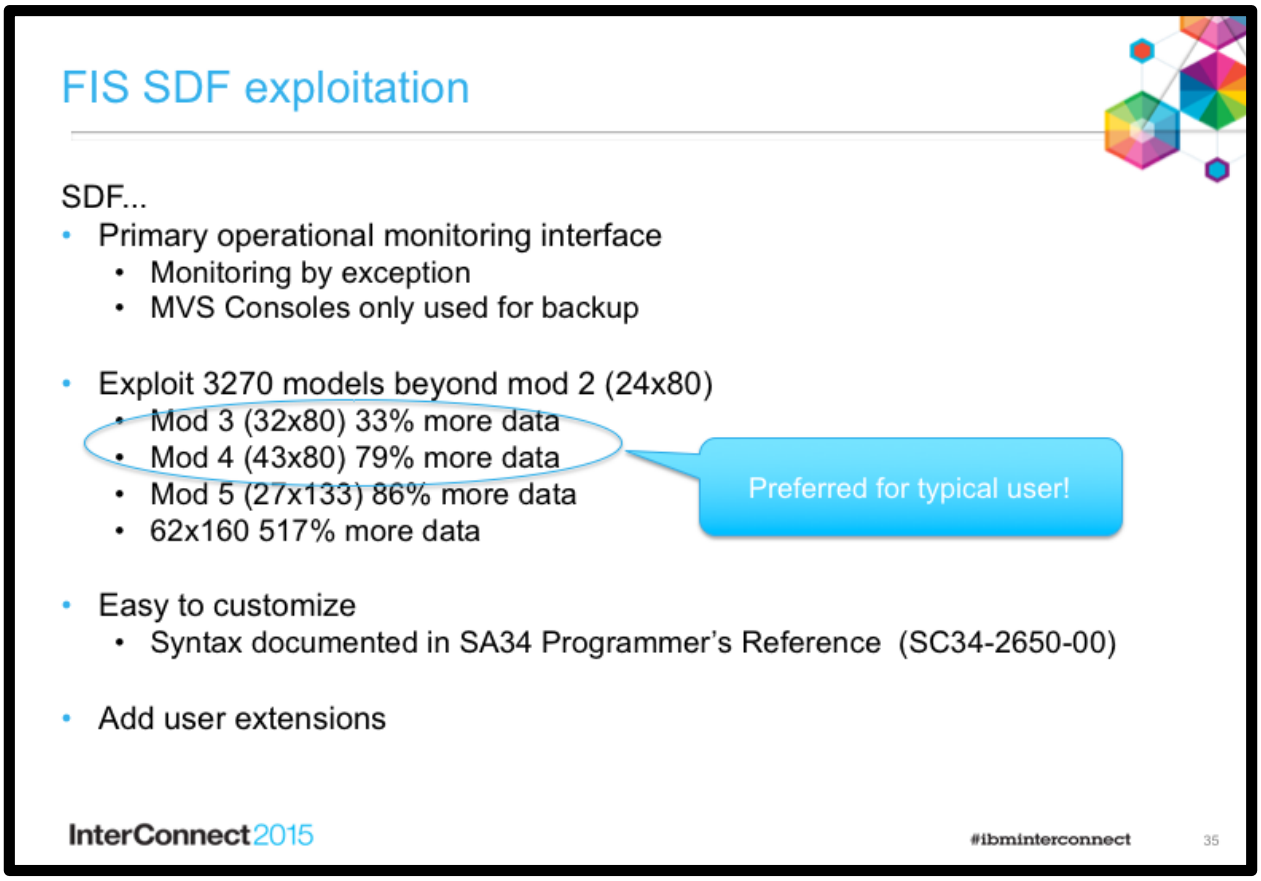
SDFTREE

Use this command, when you want to add, modify or delete a single status tree.

SDFPANEL

Use this command, when you want to add, modify or delete a single panel.

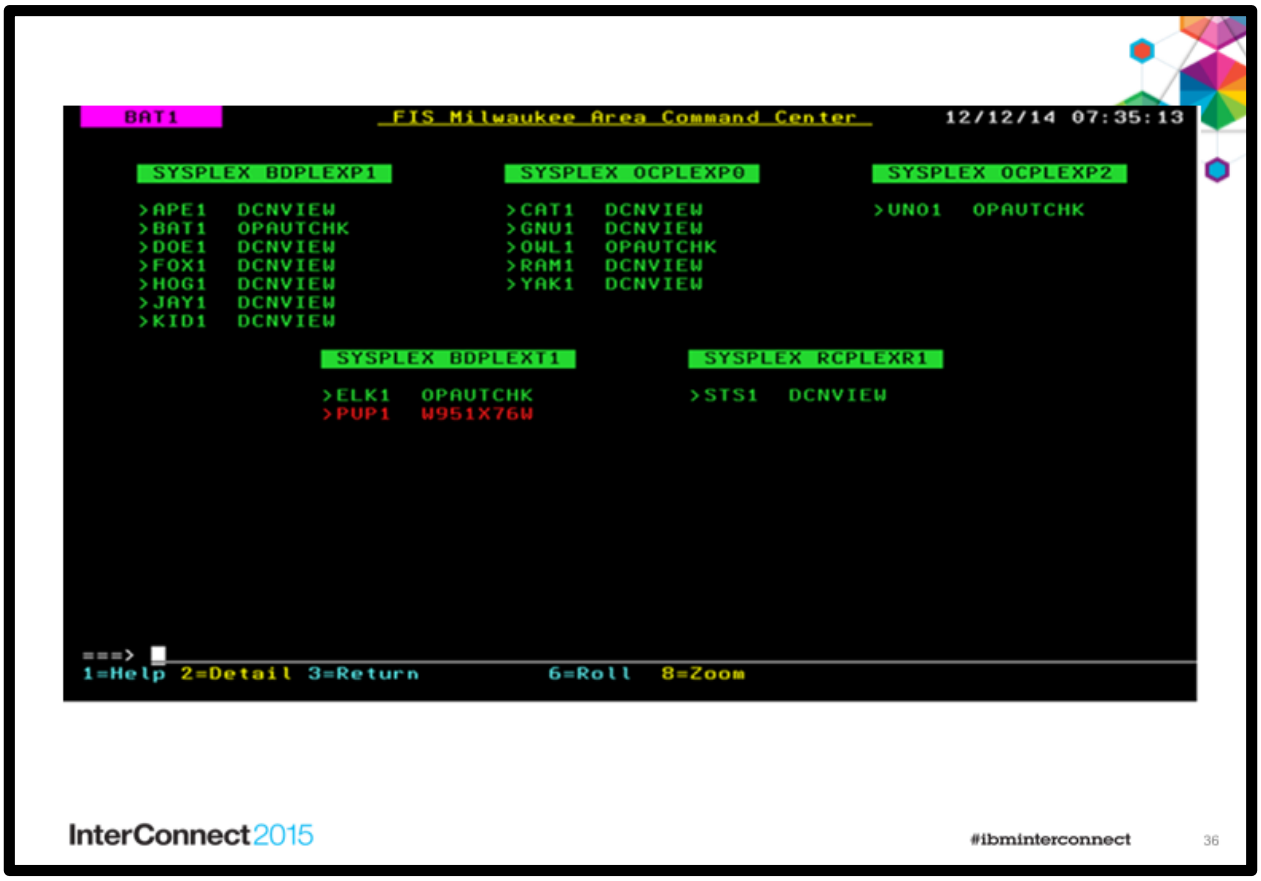

#ibminterconnect
SDF at FIS
IBM
InterConnect 2015
The Premier Cloud & Mobile Conference



I am encouraging everyone to use the higher resolution 3270 models, especially Mod 3 and Mod 4, because they offer so much more room for data and allow for maximum flexibility with unlimited panels.

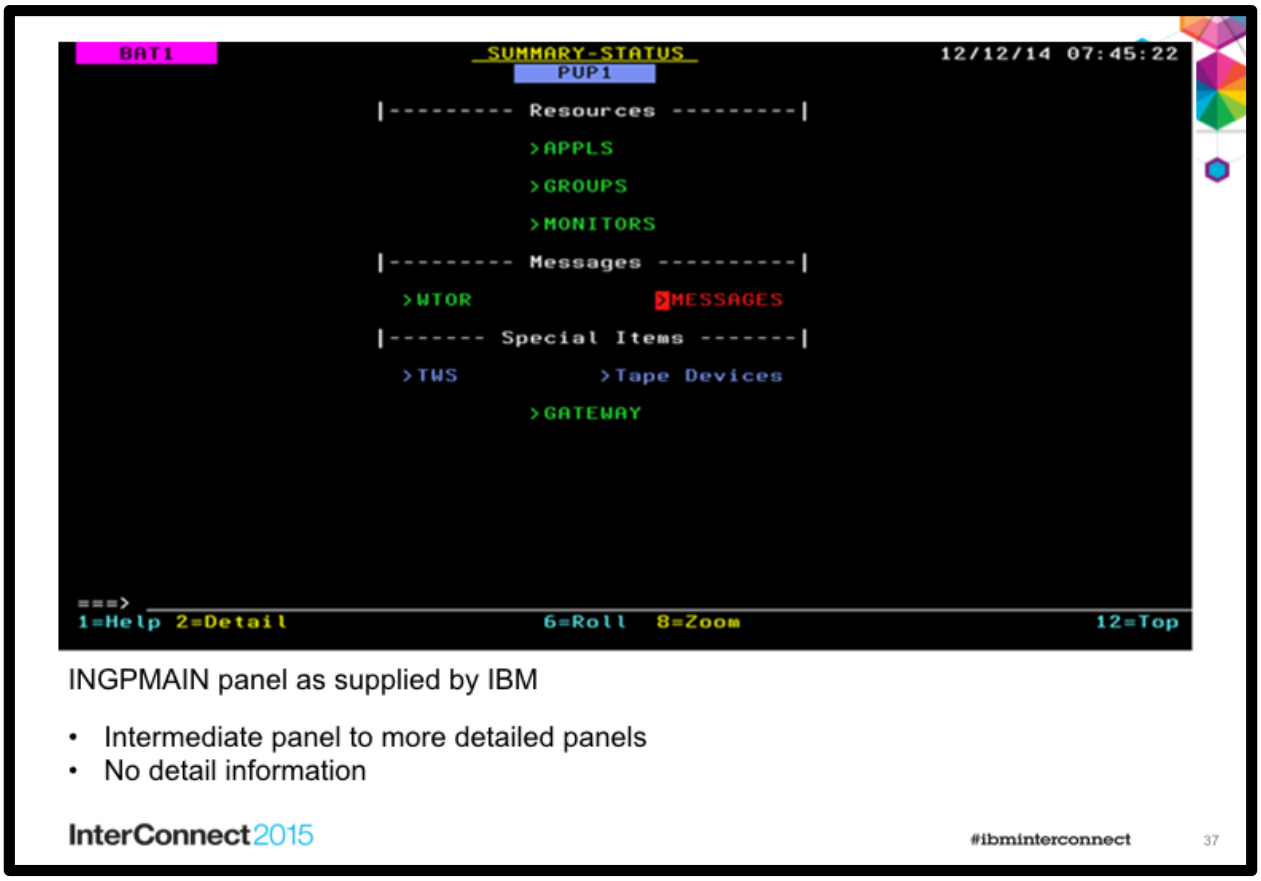
The Mod 5, while supporting more data across rows, adds only 3 rows vs. a Mod 3

The 62x160 provides the most data by far, but is limited to use on large terminals where the font does not cause eye-strain!



SDF monitoring illustration, showing the INGTOP panel customized for our BDOC datacenter

BDOC – Brown Deer Operations Center is located on our Milwaukee campus



INGPMAIN as supplied by IBM and used at our BDOC datacenter

(Note, we don't use TWS at this time)

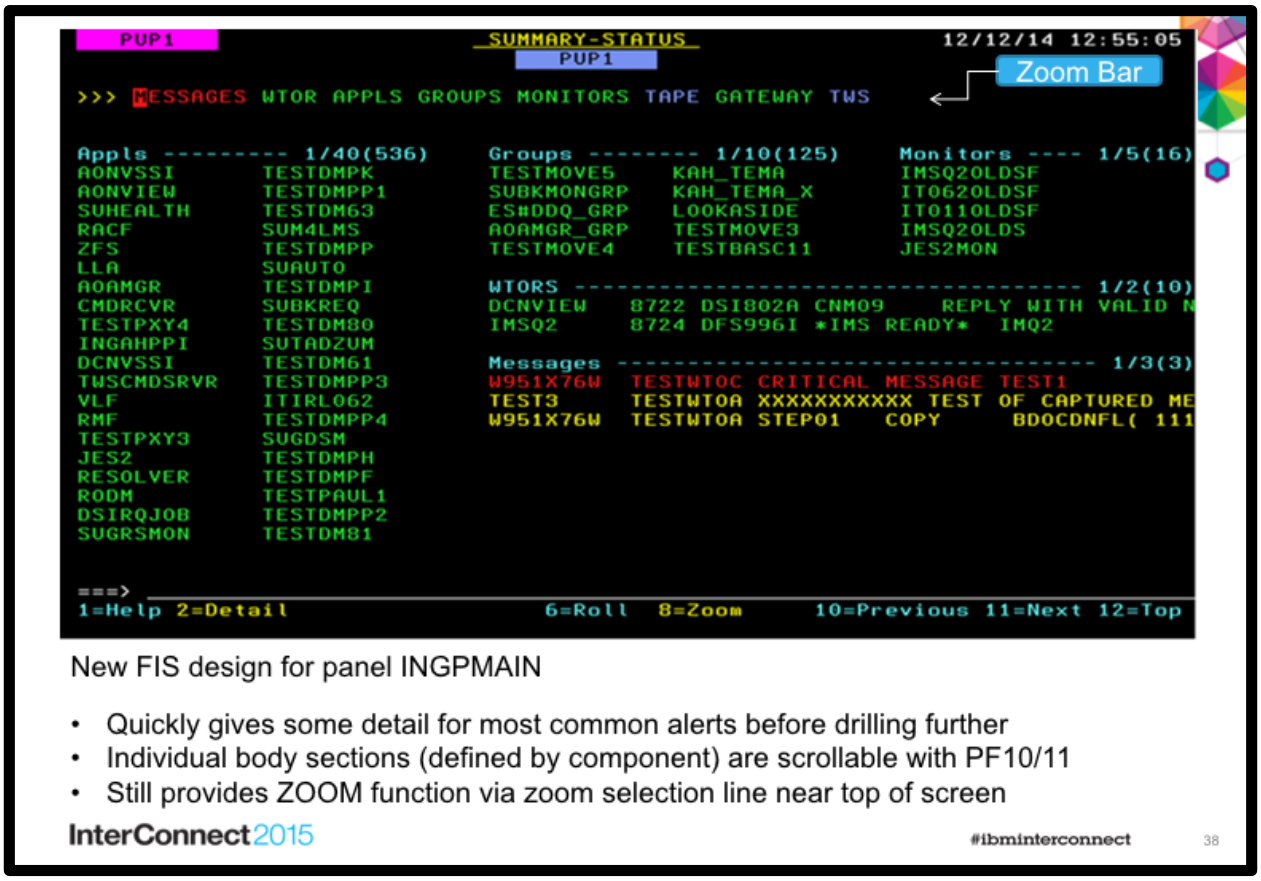
While it is an easy to use intermediate view, it does not give much detail.

Challenges:

Merging of views to satisfy two different datacenters, BDOC which uses the above and LTC which has a heavily customized view showing more detail.

Our second datacenter (LTC) view is also based on SA 3.1 technology, so it is overdue to be upgraded.

No illustrations of the LTC panel were provided for brevity.



New FIS INGPMAIN

I started with the IBM INGPMAIN panel, but merged ideas from our second (LTC) datacenter.

I optimized this screen to work well with Mod 3 (32x80), as seen here, while still supporting Mod 2 (24x80)

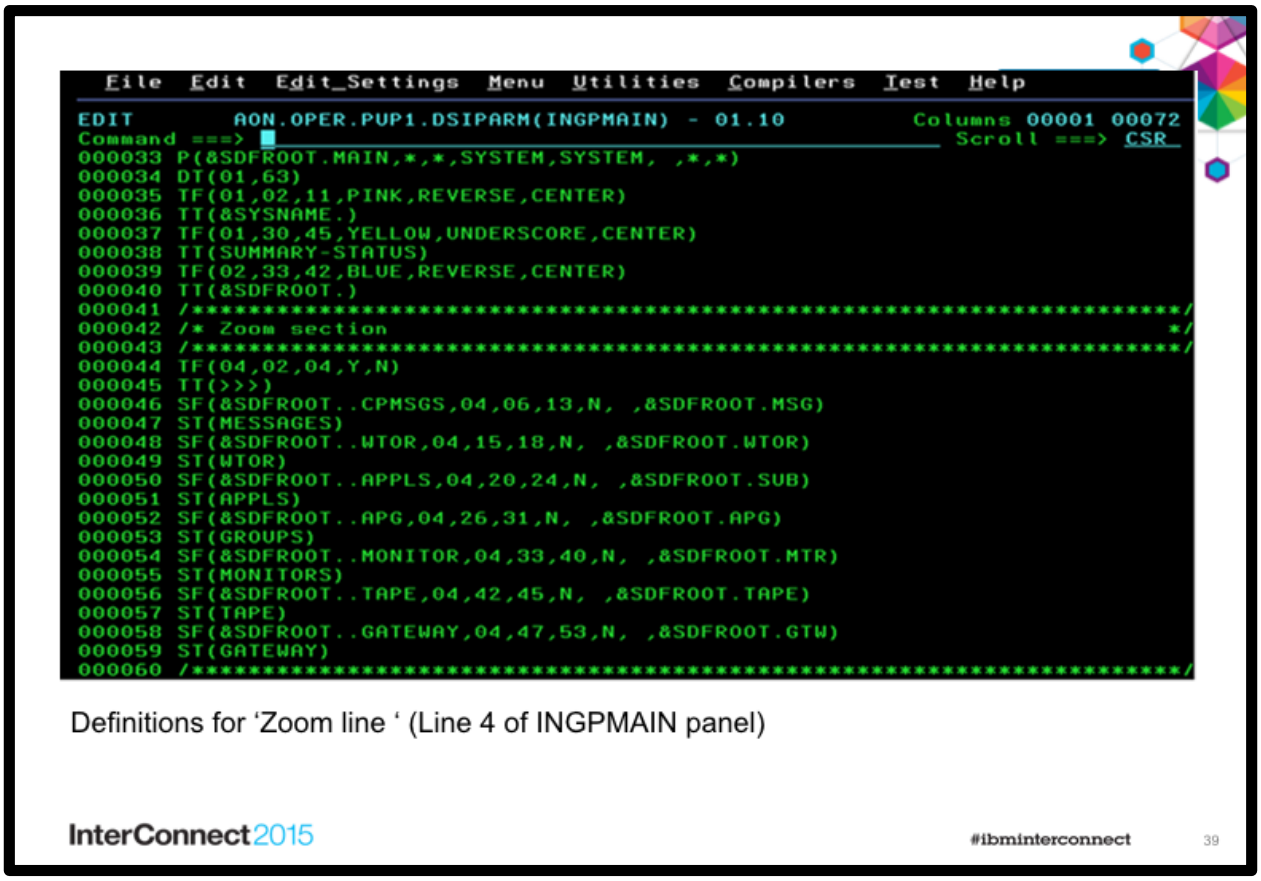
This panel makes heavy use of body and cell definitions to allow for expansion of information down and to the left for the larger screen sizes.

Near the top, you see what I call our 'Zoom Bar'. This has the same functionality as the fields on the IBM-supplied INGPMAIN, tabbing to the filed and pressing PF8 for the Zoom function.

Below that you see 5 body sections. I did not attempt to map all resource types to body sections, due to limited space. I also felt it was not necessary for all resource types, because one can still zoom to the individual resource panels. Lastly, you see TWS on the Zoom Bar, but it is not currently used (I had hopes of adapting it for Control-M alerts messages), and TAPE is all virtual, so we rarely see any alerts. However I have developed an alternative INGPMAIN which adds a TAPE Body section.

One thing not illustrated here, is our own HELP panel behind the PF1 key to assist the

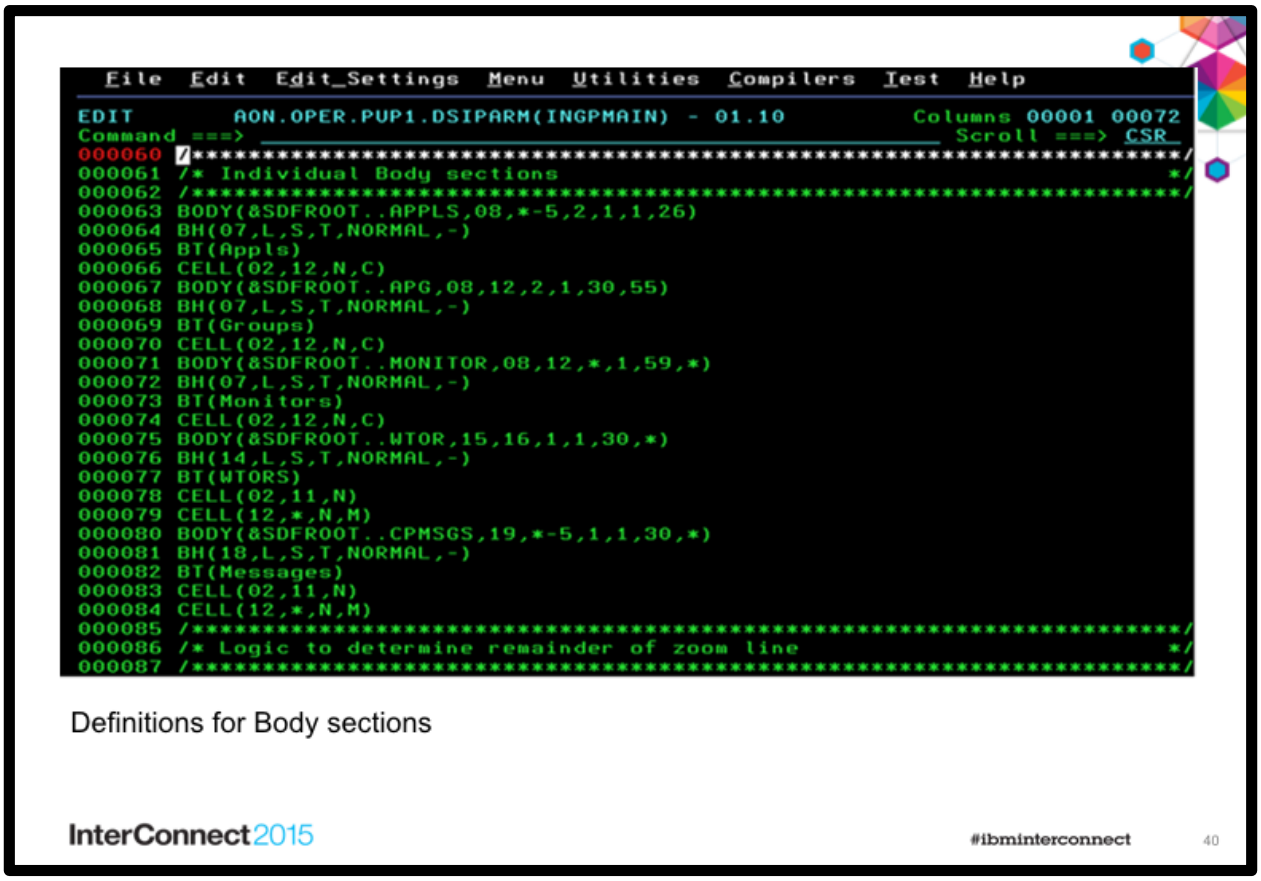
operators with the function of this panel.



This is a view of the panel definitions needed for the Zoom bar.

Simple use of the TT, SF, and ST statements to build this line

&SDFROOT is automatically populated.



This view illustrates the BODY/CELL panel definitions

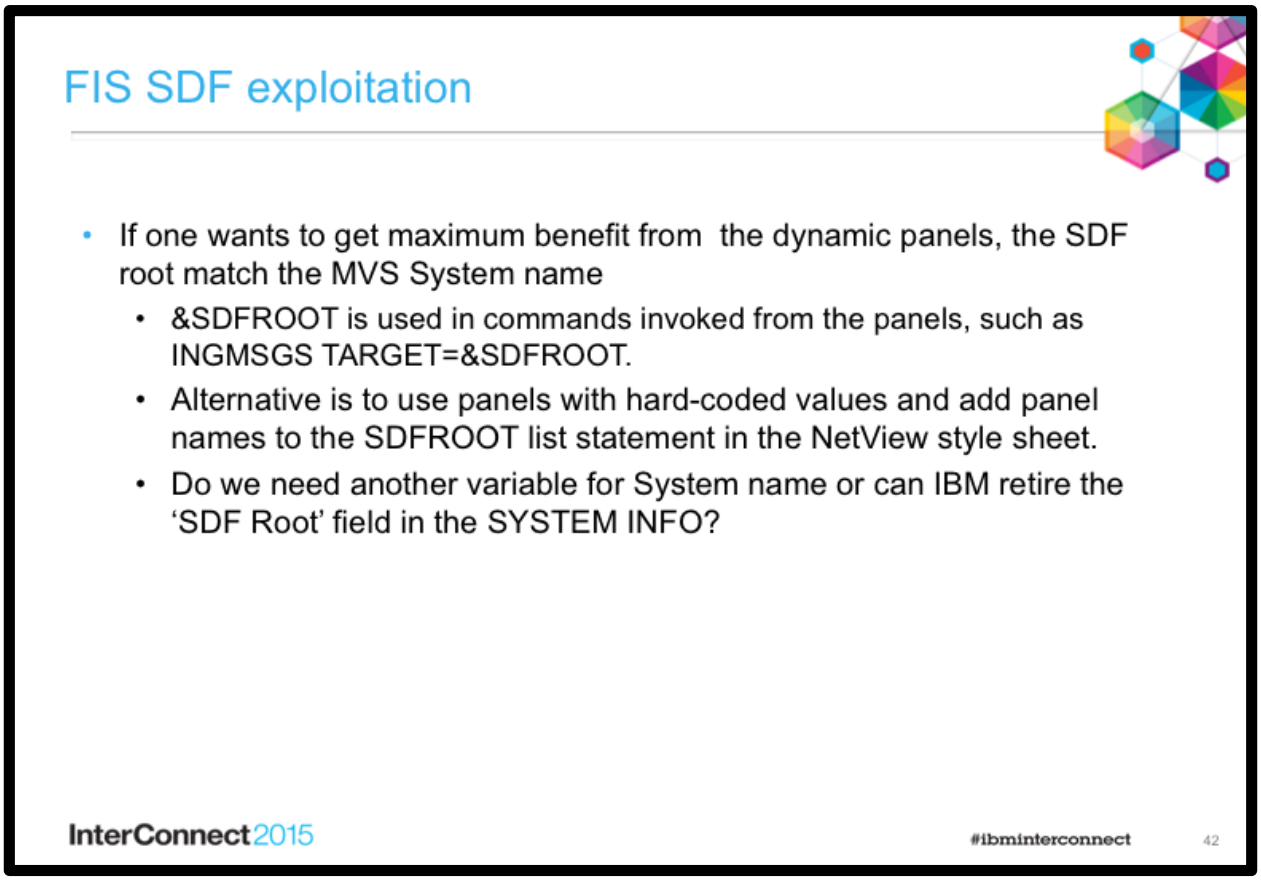
Through use of the '*-5' the Appls and Messages body sections are allowed to expand with more lines to fill larger screen sizes.

Use of an asterisk '*' for the far right column allows expansion to the right for users of MOD 5 (27x132) and the 62x160 terminal architectures.

INGMSGS illustration

This shows how to add a PFK definition to run user commands.

Here we added PF21 to launch the FIX OPDOC#B REXX exec with a hardcoded parameter ‘CPMSGS’ and appending the entire text of the message, using the #DA



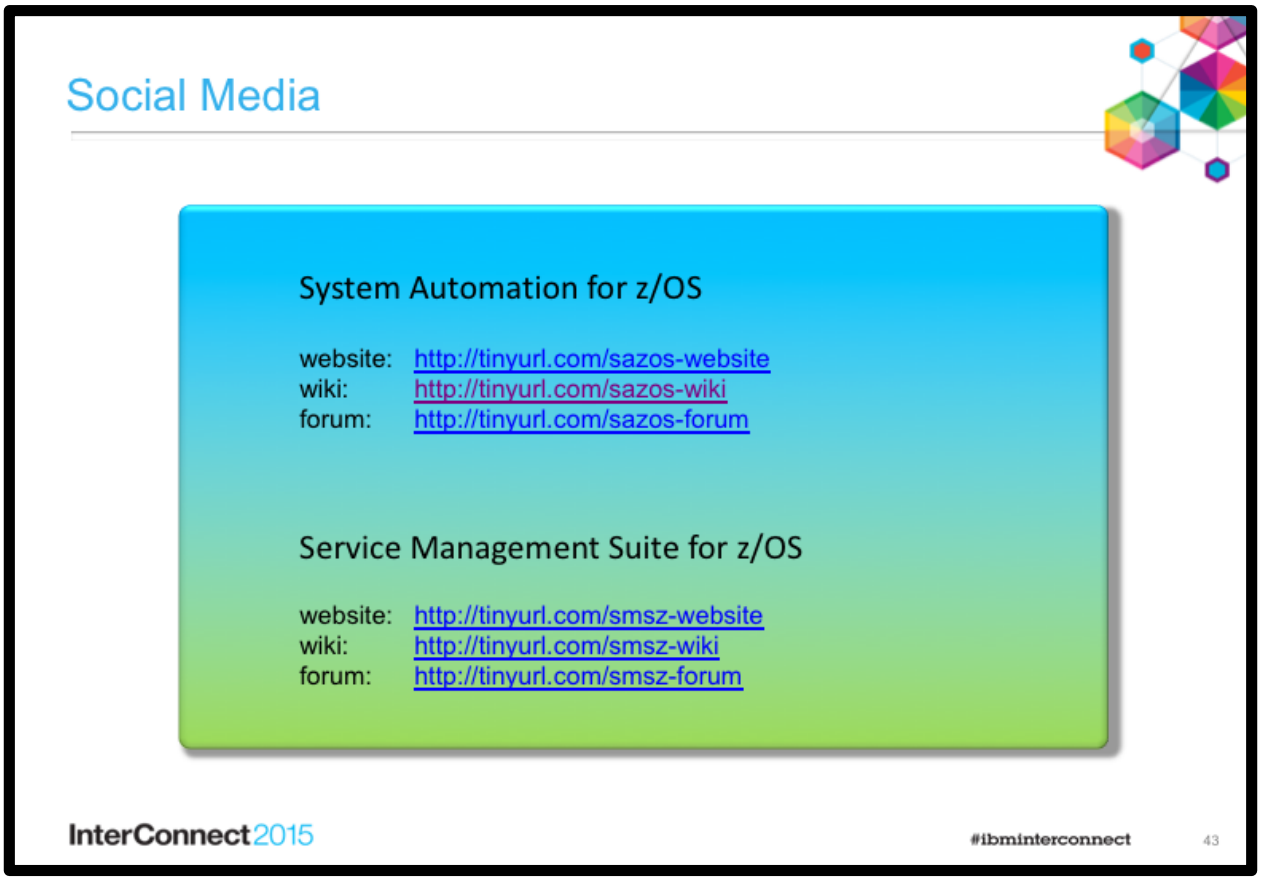
The PDB help, while indicating the system name is the default, does not mention the effect of using a different value.

Various commands defined in the SA supplied SDF panels would break.

Some miscellaneous comments:

FIS implemented SDF exit AOFEXC02, not decide what goes to SDF, but which updates could be logged to NETLOG as an audit record. Then I run a weekly job against the last week's logs, searching for the audit records, and generating a report of important alerts.

FIS has also developed some special tree and panels members to provide unique monitoring for an application requiring special monitoring of its own. I could have added it here, but that would add another 6 slides and Juergen already has the deign process covered. The only difference with this special application is I use a unique root that does not connect the application with any system.



System Automation for z/OS is a component of the Service Management Suite for z/OS. For more information about System Automation for z/OS and the Service Management Suite for z/OS, visit Service Management Connect for z Systems on developerWorks.

System Automation for z/OS social media:

Product website: http://tinyurl.com/sazos-website

Product wiki: http://tinyurl.com/sazos-wiki

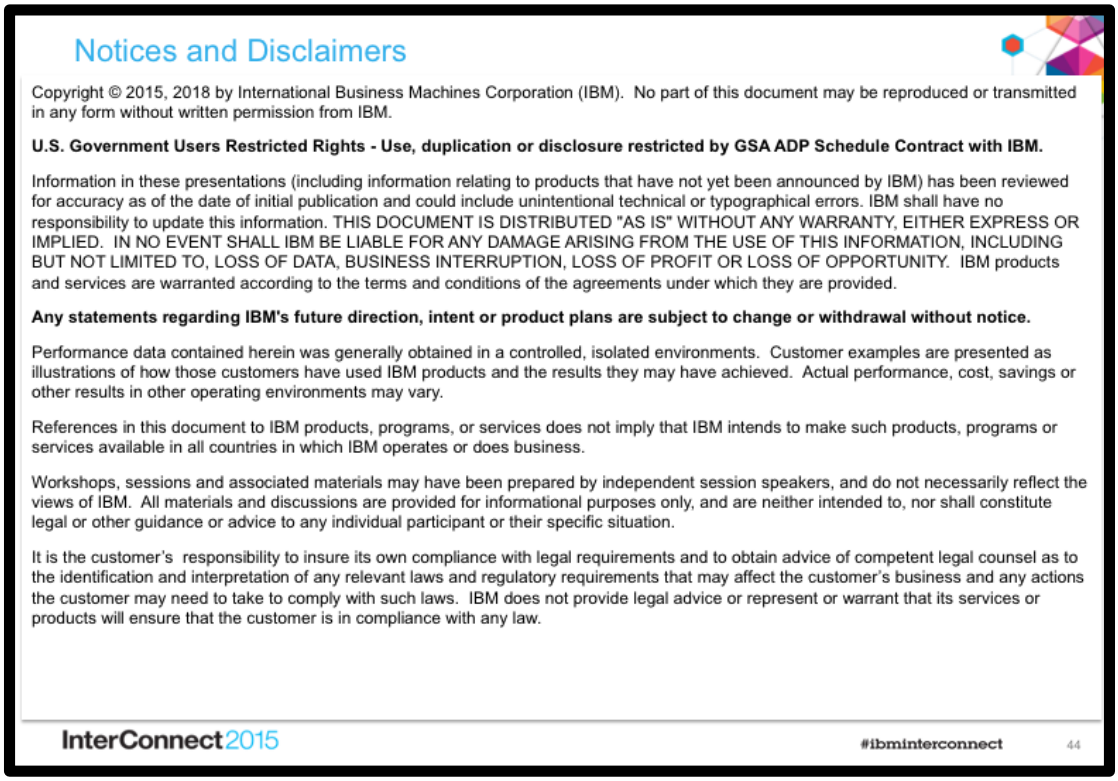
Product forum: http://tinyurl.com/sazos-forum

Service Management Suite for z/OS social media:

Product website: http://tinyurl.com/smsz-website

Product wiki: http://tinyurl.com/smsz-wiki

Product forum: http://tinyurl.com/smsz-forum



Notices and Disclaimers

Notices and Disclaimers (con't)

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products in connection with this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products. IBM does not warrant the quality of any third-party products, or the ability of any such third-party products to interoperate with IBM's products. IBM EXPRESSLY DISCLAIMS ALL WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents, copyrights, trademarks or other intellectual property right.

- IBM, the IBM logo, ibm.com, Bluemix, Blueworks Live, CICS, Clearcase, DOORS®, Enterprise Document Management System™, Global Business Services ®, Global Technology Services ®, Information on Demand, ILOG, Maximo®, MQIntegrator®, MQSeries®, Netcool®, OMEGAMON, OpenPower, PureAnalytics™, PureApplication®, pureCluster™, PureCoverage®, PureData®, PureExperience®, PureFlex®, pureQuery®, pureScale®, PureSystems®, QRadar®, Rational®, Rhapsody®, SoDA, SPSS, StoredIQ, Tivoli®, Trusteer®, urban{code}®, Watson, WebSphere®, Worklight®, X-Force® and System z® Z/OS, are trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at: www.ibm.com/legal/copytrade.shtml.

InterConnect2015 #ibminterconnect 45


#ibminterconnect
Thank You
Your Feedback is Important!
Access the InterConnect 2015 Conference CONNECT Attendee Portal to complete your session surveys from your smartphone,
laptop or conference kiosk.
IBM
InterConnect 2015
The Premier Cloud & Mobile Conference