
Subject: Re: creating documentation file

Posted by [Benjamin Hornberger](#) on Tue, 15 Mar 2005 14:45:28 GMT

[View Forum Message](#) <> [Reply to Message](#)

urkhaa@hotmail.com wrote:

- > Hello
- > I would like to create a text file which contains all the informations
- > I've put on the documentation file of my IDL procedures (all that is
- > written between ;+ and ;-).

The procedure MK_HTML_HELP (comes with IDL) creates an HTML file. I guess just extracting the documentation to a text file wouldn't be difficult in any scripting language.

Benjamin

Subject: Re: creating documentation file

Posted by [Jim Pendleton, RSI](#) on Tue, 15 Mar 2005 15:18:02 GMT

[View Forum Message](#) <> [Reply to Message](#)

<urkhaa@hotmail.com> wrote in message

news:1110897674.841908.52370@g14g2000cwa.googlegroups.com...

- > Hello
- > I would like to create a text file which contains all the informations
- > I've put on the documentation file of my IDL procedures (all that is
- > written between ;+ and ;-).
- > Does anyone of you has any script available or know on the possibility to
- > create automatically this file?

If you want more professional-looking documentation than what MK_HTML_HELP can produce, take a look at Dr. Mike Galloy's idldoc application in the RSI codebank.

<http://www.rsinc.com/codebank/search.asp?FID=100>.

Jim P.

Subject: Re: creating documentation file

Posted by [David Fanning](#) on Tue, 15 Mar 2005 15:49:05 GMT

[View Forum Message](#) <> [Reply to Message](#)

Jim Pendleton, RSI writes:

- > If you want more professional-looking documentation than what MK_HTML_HELP
- > can produce, take a look at Dr. Mike Galloy's idldoc application in the RSI

> codebank.
>
> <http://www.rsinc.com/codebank/search.asp?FID=100>.

Wow. Pretty nice. It would be great if there was an example of a marked up *.pro file, though. I get all kinds of warnings and "IDLdoc failed" messages when I run it on my files, although it does seem to produce files that are on the right track, if I just knew how to mark them up. It is possible the information is need is there, but an example might make it really clear.

Cheers,

David

--

David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: <http://www.dfanning.com/>

Subject: Re: creating documentation file
Posted by [Antonio Santiago](#) on Tue, 15 Mar 2005 16:14:23 GMT
[View Forum Message](#) <> [Reply to Message](#)

I created my own perl script:
<http://asantiago.gentelibre.org/index.php/archives/2004/12/01/a-perlscript-for-idl-documentation-generator/>

but I change to IDLdoc when I found it:
<http://asantiago.gentelibre.org/index.php/archives/2005/03/01/idldoc-documentation-utility/>

Also attach a file that use IDLdoc syntax.

```
;+
; @file_comments Product class represents a radar product to
; generate. Maintains associations with the plugins which is based
; plugin information and parent dependencies.
;
; @author Antonio Santiago
;      (santiago\@grahi.upc.edu - http://asantiago.gentelibre.org)
;
; @requires linkedlist logmsg
;
```

```

; @field parents Linkedlist of products that represnts parent or input
;       dependencies
; @field children Linkedlist of products that represent children
;       dependencies.
; @field executed Indicates if the product has been generated.
; @field name Name of the product.
; @field description Description of the product.
; @field plugin Plugin object that implements the product.
; @field plugin_info PluginInfo object that has the plugin
;       information.
;
;
; @history
;
;       Fri Jan 14 17:08:04 2005, Antonio Santiago
;       (santiago\@grahi.upc.edu - http://asantiago.gentelibre.org)
;-

```

```

PRO Product__define
  struct = { Product, $
    parents: OBJ_NEW(), $
    children: OBJ_NEW(), $

    executed: OB, $

    name: "", $
    description: "", $

    plugin: OBJ_NEW(), $ ;'Plugin' object
    plugin_info: OBJ_NEW() $ ;'PluginInfo' object
  }
END

```

```

;+
; Creates and initializes the object.
;
; @returns 1 if successful; 0 otherwise
;
; @private
;-

```

```

FUNCTION Product::Init
  self.parents = OBJ_NEW('linkedlist')
  self.children = OBJ_NEW('linkedlist')
  RETURN, 1
END

```

```

;+
; Frees the resources used by the object.

```

```

;
; @private
;-
PRO Product::Cleanup
  ;;Frees Plugin and PluginInfo objects
  OBJ_DESTROY, [self.parents, self.children, self.plugin, $
    self.plugin_info]
END

;+
; Set individual property values.
;
; @keyword name {in}{optional}{type=string} Name of the product.
; @keyword description {in}{optional}{type=string} Description of the
;       products.
;-
PRO Product::SetProperty, NAME=name, DESCRIPTION=description
  IF N_ELEMENTS(name) THEN self.name = name
  IF N_ELEMENTS(description) THEN self.description = description
END

;+
; Get individual properties.
; <i>(See SetProperty methos)</i>
;-
PRO Product::GetProperty, NAME=name, DESCRIPTION=description
  IF ARG_PRESENT(name) THEN name = self.name
  IF ARG_PRESENT(description) THEN description = self.description
END

;+
; Sets a parent dependency for the object and automatically sets a
; reference from parent to child.
;
; @param product {in}{required}{type=Product} Product object that
;       represents a parent dependency.
;-
PRO Product::AddParent, product
  IF NOT OBJ_ISA(product, 'Product') THEN BEGIN
    logmsg, 'ERROR', 'The object is not a Product object !!!'
    RETURN
  ENDIF
  self.parents->Add, product
END

;+

```

```

; Sets a child dependency for the object and set a reference from
; child to parent.
;
; @param product {in}{required}{type=Product} Product object that
;     represents a child dependency.
;-
PRO Product::AddChild, product
    IF NOT OBJ_ISA(product, 'Product') THEN BEGIN
        logmsg, 'ERROR', 'The object is not a PRODUCT object !!!'
        RETURN
    ENDIF
    self.children->Add, product
END

```

```

;+
; Gets the parent dependencies of the product
;
; @returns Linkedlist of Product.
;-
FUNCTION Product::GetParents
    RETURN, self.parents
END

```

```

;+
; Gets the children referencies.
;
; @returns Linkedlist of Products
;-
FUNCTION Product::GetChildren
    RETURN, self.children
END

```

```

;+
; Executes the plugin associated with the Product to generate the
; output data.
;-
PRO Product::MakeProduct

    ;;Test if it was executed yet
    IF self.executed THEN RETURN

    ;;Test if all parents are executed before.
    num = self.parents->Get_Count()
    executed = 1
    i = 0

```

```

WHILE i LT num AND executed EQ 1 DO BEGIN
    product = self.parents->Get_Item(i, /DEREFERENCE)
    executed = product.executed
    i++
ENDWHILE
IF executed NE 1 THEN RETURN

;;Collect the input parameters for the Product.
input_params = self.plugin_info->CollectInputData()
self.plugin->SetInputData, input_params

;;The node is not executed yet. We execute it and all its children
logmsg, 'DEBUG', 'Go to execute the plugin of the product: ' + self.name
result = self.plugin->Execute()
logmsg, 'DEBUG', 'Result was: ' + STRING(result)

;;Test if there was errors.
;;If all is ok then mars as executed, otherwise returns and
;;don't execute its children.
IF result EQ 1 THEN self.executed = 1 $
ELSE RETURN

num = self.children->Get_Count()
FOR i=0, num-1 DO BEGIN
    product = self.children->Get_Item(i, /DEREFERENCE)
    product->MakeProduct
ENDFOR
END

;+
; Sets the Plugin object on which is based the Product.
;
; @param plugin {in}{required}{type=Plugin} Plugin on which is
; related.
;-
PRO Product::SetPlugin, plugin
    IF NOT OBJ_ISA(plugin, 'Plugin') THEN BEGIN
        logmsg, 'ERROR', 'The object is not a PLUGIN object !!!'
        RETURN
    ENDIF
    self.plugin = plugin
END

;+
; Sets the PluginInfo object that stores the Plugin information.
;

```

```

; @param plugin_info {in}{required}{type=PluginInfo} PluginInfo object.
;-
PRO Product::SetPluginInfo, plugin_info
  IF NOT OBJ_ISA(plugin_info, 'PluginInfo') THEN BEGIN
    logmsg, 'ERROR', 'The object is not a PLUGIN_INFO object !!!'
    RETURN
  ENDIF
  self.plugin_info = plugin_info
END

```

```

;+
; Creates an association from PluginInfo to Plugin object of the
; Product.
;
; @pre Product object must has references to a valid Plugin and
; PluginInfo objects.
;-
PRO Product::AssocPluginInfo2Plugin
  IF OBJ_ISA(self.plugin, 'Plugin') AND $
    OBJ_ISA(self.plugin_info, 'PluginInfo') THEN BEGIN
    self.plugin_info->SetPlugin, self.plugin
  ENDIF
END

```

```

;+
; Gets the Plugin object on which the Product is based.
;
; @returns Plugin
;-
FUNCTION Product::GetPlugin
  RETURN, self.plugin
END

```

```

;+
; Gets the PluginInfo object with the information of the Plugin on
; which the Product is based.
;
; @returns PluginInfo
;-
FUNCTION Product::GetPluginInfo
  RETURN, self.plugin_info
END

```

File Attachments

1) [product__define.pro](#), downloaded 99 times

dependencies.

</TD>

</TR>

<TR>

<TD ALIGN="right" VALIGN="top" WIDTH="1%">

EXECUTED

<NOBR>

byte

</NOBR>

</TD>

<TD VALIGN="top">

Indicates if the product has been generated.

</TD>

</TR>

<TR>

<TD ALIGN="right" VALIGN="top" WIDTH="1%">

NAME

<NOBR>

string

</NOBR>

</TD>

<TD VALIGN="top">

Name of the product.

</TD>

</TR>

<TR>

<TD ALIGN="right" VALIGN="top" WIDTH="1%">

DESCRIPTION

<NOBR>

string

</NOBR>

</TD>

<TD VALIGN="top">

Description of the product.

</TD>

</TR>

<TR>

<TD ALIGN="right" VALIGN="top" WIDTH="1%">

PLUGIN

<NOBR>

object reference

</NOBR>

</TD>

<TD VALIGN="top">

Plugin object that implements the product.

</TD>

</TR>

<TR>

```

<TD ALIGN="right" VALIGN="top" WIDTH="1%">
<FONT CLASS="param_name">PLUGIN_INFO</FONT><BR>
<FONT CLASS="param_attrib"><NOBR>
object reference
</NOBR></FONT>
</TD>

```

```

<TD VALIGN="top">
  PluginInfo object that has the plugin
  information.

```

```

</TD>
</TR>
</TABLE>

```

```

<P>
<TABLE CELLSPACING="3" CELLPADDING="0" CLASS="listing">

```

```

<TR><TD CLASS="title">
<A NAME="_routine_summary">Routine Summary</A>
&nbsp;<FONT CLASS="list_tagline">15 routines</FONT>
</TD></TR>

```

```

<TABLE CELLSPACING="2" CELLPADDING="0" BORDER="0" STYLE="BORDER-BOTTOM:
1px groove" WIDTH="100%"><TR><TD ALIGN="left" VALIGN="top"><P
CLASS="code_header">procedure <A
HREF="#_Product__define">Product__define</A></P></TD><TD ALIGN="right"
VALIGN="top"></TD><TR><TD><P class="first_line"></P></TD></TR></TABLE>

```

```

<TABLE CELLSPACING="2" CELLPADDING="0" BORDER="0" STYLE="BORDER-BOTTOM:
1px groove" WIDTH="100%"><TR><TD ALIGN="left" VALIGN="top"><P
CLASS="code_header">function <A HREF="#_Product::Init">Product::Init</A>()</P></TD><TD
ALIGN="right" VALIGN="top"><FONT CLASS="param_attrib">private
</FONT></TD><TR><TD><P class="first_line"> Creates and initializes the
object.</P></TD></TR></TABLE>

```

```

<TABLE CELLSPACING="2" CELLPADDING="0" BORDER="0" STYLE="BORDER-BOTTOM:
1px groove" WIDTH="100%"><TR><TD ALIGN="left" VALIGN="top"><P
CLASS="code_header">procedure <A
HREF="#_Product::Cleanup">Product::Cleanup</A></P></TD> <TD ALIGN="right"
VALIGN="top"><FONT CLASS="param_attrib">private </FONT></TD><TR><TD><P
class="first_line"> Frees the resources used by the object.</P></TD></TR></TABLE>

```

```

<TABLE CELLSPACING="2" CELLPADDING="0" BORDER="0" STYLE="BORDER-BOTTOM:
1px groove" WIDTH="100%"><TR><TD ALIGN="left" VALIGN="top"><P
CLASS="code_header">procedure <A
HREF="#_Product::SetProperty">Product::SetProperty</A>, [NAME=<EM>string</EM>],
[DESCRIPTION=<EM>string</EM>]</P></TD><TD ALIGN="right"
VALIGN="top"></TD><TR><TD><P class="first_line"> Set individual property
values.</P></TD></TR></TABLE>

```

```

<TABLE CELLSPACING="2" CELLPADDING="0" BORDER="0" STYLE="BORDER-BOTTOM:
1px groove" WIDTH="100%"><TR><TD ALIGN="left" VALIGN="top"><P
CLASS="code_header">procedure <A
HREF="#_Product::GetProperty">Product::GetProperty</A>, NAME=<EM>NAME</EM>,
DESCRIPTION=<EM>DESCRIPTION</EM></P></TD><TD ALIGN="right"

```

```

VALIGN="top"></TD><TR><TD><P class="first_line"> Get individual
properties.</P></TD></TR></TABLE>
<TABLE CELLSPACING="2" CELLPADDING="0" BORDER="0" STYLE="BORDER-BOTTOM:
1px groove" WIDTH="100%"><TR><TD ALIGN="left" VALIGN="top"><P
CLASS="code_header">procedure <A HREF="#_Product::AddParent">Product::AddParent</A>,
product</P></TD><TD ALIGN="right" VALIGN="top"></TD><TR><TD><P class="first_line">
Sets a parent dependency for the object and automatically sets a reference from parent to
child.</P></TD></TR></TABLE>
<TABLE CELLSPACING="2" CELLPADDING="0" BORDER="0" STYLE="BORDER-BOTTOM:
1px groove" WIDTH="100%"><TR><TD ALIGN="left" VALIGN="top"><P
CLASS="code_header">procedure <A HREF="#_Product::AddChild">Product::AddChild</A>,
product</P></TD><TD ALIGN="right" VALIGN="top"></TD><TR><TD><P class="first_line">
Sets a child dependency for the object and set a reference from child to
parent.</P></TD></TR></TABLE>
<TABLE CELLSPACING="2" CELLPADDING="0" BORDER="0" STYLE="BORDER-BOTTOM:
1px groove" WIDTH="100%"><TR><TD ALIGN="left" VALIGN="top"><P
CLASS="code_header">function <A
HREF="#_Product::GetParents">Product::GetParents</A>()</P ></TD><TD ALIGN="right"
VALIGN="top"></TD><TR><TD><P class="first_line"> Gets the parent dependencies of the
product </P></TD></TR></TABLE>
<TABLE CELLSPACING="2" CELLPADDING="0" BORDER="0" STYLE="BORDER-BOTTOM:
1px groove" WIDTH="100%"><TR><TD ALIGN="left" VALIGN="top"><P
CLASS="code_header">function <A
HREF="#_Product::GetChildren">Product::GetChildren</A>() </P></TD><TD ALIGN="right"
VALIGN="top"></TD><TR><TD><P class="first_line"> Gets the children
referencies.</P></TD></TR></TABLE>
<TABLE CELLSPACING="2" CELLPADDING="0" BORDER="0" STYLE="BORDER-BOTTOM:
1px groove" WIDTH="100%"><TR><TD ALIGN="left" VALIGN="top"><P
CLASS="code_header">procedure <A
HREF="#_Product::MakeProduct">Product::MakeProduct</A></P ></TD><TD ALIGN="right"
VALIGN="top"></TD><TR><TD><P class="first_line"> Executes the plugin associated with the
Product to generate the output data.</P></TD></TR></TABLE>
<TABLE CELLSPACING="2" CELLPADDING="0" BORDER="0" STYLE="BORDER-BOTTOM:
1px groove" WIDTH="100%"><TR><TD ALIGN="left" VALIGN="top"><P
CLASS="code_header">procedure <A HREF="#_Product::SetPlugin">Product::SetPlugin</A>,
plugin</P></TD><TD ALIGN="right" VALIGN="top"></TD><TR><TD><P class="first_line"> Sets
the Plugin object on which is based the Product.</P></TD></TR></TABLE>
<TABLE CELLSPACING="2" CELLPADDING="0" BORDER="0" STYLE="BORDER-BOTTOM:
1px groove" WIDTH="100%"><TR><TD ALIGN="left" VALIGN="top"><P
CLASS="code_header">procedure <A
HREF="#_Product::SetPluginInfo">Product::SetPluginInfo</A >, plugin_info</P></TD><TD
ALIGN="right" VALIGN="top"></TD><TR><TD><P class="first_line"> Sets the PluginInfo object
that stores the Plugin information.</P></TD></TR></TABLE>
<TABLE CELLSPACING="2" CELLPADDING="0" BORDER="0" STYLE="BORDER-BOTTOM:
1px groove" WIDTH="100%"><TR><TD ALIGN="left" VALIGN="top"><P
CLASS="code_header">procedure <A
HREF="#_Product::AssocPluginInfo2Plugin">Product::AssocPluginInfo2Plugin
</A></P></TD><TD ALIGN="right" VALIGN="top"></TD><TR><TD><P class="first_line">

```

Creates an association from PluginInfo to Plugin object of the

Product.</P></TD></TR></TABLE>

<TABLE CELLSPACING="2" CELLPADDING="0" BORDER="0" STYLE="BORDER-BOTTOM:

1px groove" WIDTH="100%"><TR><TD ALIGN="left" VALIGN="top"><P

CLASS="code_header">function Product::GetPlugin()</P>

></TD><TD ALIGN="right" VALIGN="top"></TD><TR><TD><P class="first_line"> Gets the

Plugin object on which the Product is based.</P></TD></TR></TABLE>

<TABLE CELLSPACING="2" CELLPADDING="0" BORDER="0" STYLE="BORDER-BOTTOM:

1px groove" WIDTH="100%"><TR><TD ALIGN="left" VALIGN="top"><P

CLASS="code_header">function Product::GetPluginInfo()</P></TD><TD ALIGN="right"

VALIGN="top"></TD><TR><TD><P class="first_line"> Gets the PluginInfo object with the

information of the Plugin on which the Product is based.</P></TD></TR></TABLE>

</TABLE>

<P>

<TABLE CELLPADDING="3" CELLSPACING="0" CLASS="listing">

<TR><TD CLASS="title">Routine Details</TD></TR>

</TABLE>

<P>

<TABLE WIDTH="100%" BORDER="0"><TR><TD>

<H3 CLASS="routine">

Product__define

</H3>

</TD><TD ALIGN="right">

</TD></TR></TABLE>

<TABLE CELLSPACING="2" CELLPADDING="0" BORDER="0" WIDTH="100%"><TR><TD

ALIGN="left" VALIGN="top"><P CLASS="code_header">procedure

Product__define</P></TD></TR></TABLE>

<P>

<P>

<DL>

<DT CLASS="attribute">Requires

<DD CLASS="attribute_item">

linkedlist logmsg

</DD></DT>

<DT CLASS="attribute">History

<DD CLASS="attribute_item">

Fri Jan 14 17:08:04 2005, Antonio Santiago

(santiago@grahi.upc.edu - http://asantiago.gentilibre.org)

</DD></DT>

<DT CLASS="attribute">Author

<DD CLASS="attribute_item">

Antonio Santiago

```
</DD></DT>
</DL>
<P>
<P>
<P>
<HR CLASS="divider">
<TABLE WIDTH="100%" BORDER="0"><TR><TD>
<H3 CLASS="routine"><A NAME="_Product::Init">
Product::Init
</A></H3>
</TD><TD ALIGN="right">
<FONT CLASS="param_attrib">private </FONT>
</TD></TR></TABLE>
<TABLE CELLSPACING="2" CELLPADDING="0" BORDER="0" WIDTH="100%"><TR><TD
ALIGN="left" VALIGN="top"><P CLASS="code_header">function
Product::Init()</P></TD></TR></TABLE>
<P>
```

Creates and initializes the object.

```
<P>
<DL>
<DT CLASS="attribute">Returns
<DD CLASS="attribute_item">
1 if successful; 0 otherwise
```

```
</DD></DT>
</DL>
<P>
<P>
<P>
<HR CLASS="divider">
<TABLE WIDTH="100%" BORDER="0"><TR><TD>
<H3 CLASS="routine"><A NAME="_Product::Cleanup">
Product::Cleanup
</A></H3>
</TD><TD ALIGN="right">
<FONT CLASS="param_attrib">private </FONT>
</TD></TR></TABLE>
<TABLE CELLSPACING="2" CELLPADDING="0" BORDER="0" WIDTH="100%"><TR><TD
ALIGN="left" VALIGN="top"><P CLASS="code_header">procedure
Product::Cleanup</P></TD></TR></TABLE>
<P>
```

Frees the resources used by the object.

```

<P>
<P>
<P>
<HR CLASS="divider">
<TABLE WIDTH="100%" BORDER="0"><TR><TD>
<H3 CLASS="routine"><A NAME="_Product::SetProperty">
Product::SetProperty
</A></H3>
</TD><TD ALIGN="right">

</TD></TR></TABLE>
<TABLE CELLSPACING="2" CELLPADDING="0" BORDER="0" WIDTH="100%"><TR><TD
ALIGN="left" VALIGN="top"><P CLASS="code_header">procedure Product::SetProperty, [<A
HREF=#_Product::SetProperty_keyword_NAME>NAME</A>=<EM>string </EM>], [<A
HREF=#_Product::SetProperty_keyword_DESCRIPTION>DESCRIPTION
</A>=<EM>string</EM>]</P></TD></TR></TABLE>
<P>

```

Set individual property values.

```

<P>
<P>
<TABLE CELLPADDING="3" CELLSPACING="0" CLASS="parameters">
<TR BGCOLOR="#EEEEFF" CLASS="small_title">
<TD COLSPAN=2>Keywords</TD>
</TR>
<TR BGCOLOR="white" CLASS="parameters">
<TD ALIGN="right" VALIGN="top" WIDTH="1%">
<A NAME="_Product::SetProperty_keyword_NAME"/>
<FONT CLASS="param_name">NAME</FONT><BR>
<FONT CLASS="param_attrib">
<NOBR>in, optional</NOBR><BR>
<NOBR>string</NOBR>
</FONT>
</TD>
<TD VALIGN="top">
Name of the product.
</TD>
</TR>
<TR BGCOLOR="white" CLASS="parameters">
<TD ALIGN="right" VALIGN="top" WIDTH="1%">
<A NAME="_Product::SetProperty_keyword_DESCRIPTION"/>
<FONT CLASS="param_name">DESCRIPTION</FONT><BR>
<FONT CLASS="param_attrib">
<NOBR>in, optional</NOBR><BR>
<NOBR>string</NOBR>
</FONT>
</TD>

```

```
<TD VALIGN="top">
  Description of the
      products.
```

```
</TD>
```

```
</TR>
```

```
</TABLE>
```

```
<P>
```

```
<HR CLASS="divider">
```

```
<TABLE WIDTH="100%" BORDER="0"><TR><TD>
```

```
<H3 CLASS="routine"><A NAME="_Product::GetProperty">
```

```
Product::GetProperty
```

```
</A></H3>
```

```
</TD><TD ALIGN="right">
```

```
</TD></TR></TABLE>
```

```
<TABLE CELLSPACING="2" CELLPADDING="0" BORDER="0" WIDTH="100%"><TR><TD
ALIGN="left" VALIGN="top"><P CLASS="code_header">procedure Product::GetProperty, <A
HREF=#_Product::GetProperty_keyword_NAME>NAME</A>=<EM>NAME </EM>, <A
HREF=#_Product::GetProperty_keyword_DESCRIPTION>DESCRIPTION
```

```
</A>=<EM>DESCRIPTION</EM></P></TD></TR></TABLE>
```

```
<P>
```

Get individual properties.

<i>(See SetProperty methos)</i>

```
<P>
```

```
<P>
```

```
<TABLE CELLPADDING="3" CELLSPACING="0" CLASS="parameters">
```

```
<TR BGCOLOR="#EEEEFF" CLASS="small_title">
```

```
<TD COLSPAN=2>Keywords</TD>
```

```
</TR>
```

```
<TR BGCOLOR="white" CLASS="parameters">
```

```
<TD ALIGN="right" VALIGN="top" WIDTH="1%">
```

```
<A NAME="_Product::GetProperty_keyword_NAME"/>
```

```
<FONT CLASS="param_name">NAME</FONT><BR>
```

```
<FONT CLASS="param_attrib">
```

```
</FONT>
```

```
</TD>
```

```
<TD VALIGN="top">
```

```
.
```

```
</TD>
```

```
</TR>
```

```
<TR BGCOLOR="white" CLASS="parameters">
```

```
<TD ALIGN="right" VALIGN="top" WIDTH="1%">
```

```
<A NAME="_Product::GetProperty_keyword_DESCRIPTION"/>
```

```
<FONT CLASS="param_name">DESCRIPTION</FONT><BR>
```

```
<FONT CLASS="param_attrib">
```

```
</FONT>
```

```
</TD>
```

```

<TD VALIGN="top">
.
</TD>
</TR>
</TABLE>
<P>
<HR CLASS="divider">
<TABLE WIDTH="100%" BORDER="0"><TR><TD>
<H3 CLASS="routine"><A NAME="_Product::AddParent">
Product::AddParent
</A></H3>
</TD><TD ALIGN="right">

</TD></TR></TABLE>
<TABLE CELLPADDING="2" CELLSPACING="0" BORDER="0" WIDTH="100%"><TR><TD
ALIGN="left" VALIGN="top"><P CLASS="code_header">procedure Product::AddParent, <A
HREF=#_Product::AddParent_param_product>product</A></P></TD ></TR></TABLE>
<P>

```

Sets a parent dependency for the object and automatically sets a reference from parent to child.

```

<P>
<TABLE CELLPADDING="3" CELLSPACING="0" CLASS="parameters">
<TR BGCOLOR="#EEEEFF" CLASS="small_title">
<TD COLSPAN=2>Parameters</TD>
</TR>
<TR BGCOLOR="white" CLASS="parameters">
<TD ALIGN="right" VALIGN="top" WIDTH="1%">
<A NAME="_Product::AddParent_param_product"/>
<FONT CLASS="param_name">product</FONT><BR>
<FONT CLASS="param_attrib">
<NOBR>in, required</NOBR><BR>
<NOBR>Product</NOBR>
</FONT>
</TD>
<TD VALIGN="top">
Product object that
        represents a parent dependency.
</TD>
</TR>
</TABLE>
<P>
<P>
<HR CLASS="divider">
<TABLE WIDTH="100%" BORDER="0"><TR><TD>
<H3 CLASS="routine"><A NAME="_Product::AddChild">
Product::AddChild

```

```
</A></H3>
</TD><TD ALIGN="right">
```

```
</TD></TR></TABLE>
<TABLE CELLSPACING="2" CELLPADDING="0" BORDER="0" WIDTH="100%"><TR><TD
ALIGN="left" VALIGN="top"><P CLASS="code_header">procedure Product::AddChild, <A
HREF=#_Product::AddChild_param_product>product</A></P></TD ></TR></TABLE>
<P>
```

Sets a child dependency for the object and set a reference from child to parent.

```
<P>
<TABLE CELLPADDING="3" CELLSPACING="0" CLASS="parameters">
<TR BGCOLOR="#EEEEFF" CLASS="small_title">
<TD COLSPAN=2>Parameters</TD>
</TR>
<TR BGCOLOR="white" CLASS="parameters">
<TD ALIGN="right" VALIGN="top" WIDTH="1%">
<A NAME="_Product::AddChild_param_product"/>
<FONT CLASS="param_name">product</FONT><BR>
<FONT CLASS="param_attrib">
<NOBR>in, required</NOBR><BR>
<NOBR>Product</NOBR>
</FONT>
</TD>
<TD VALIGN="top">
Product object that
        represents a child dependency.
```

```
</TD>
</TR>
</TABLE>
<P>
<P>
<HR CLASS="divider">
<TABLE WIDTH="100%" BORDER="0"><TR><TD>
<H3 CLASS="routine"><A NAME="_Product::GetParents">
Product::GetParents
</A></H3>
</TD><TD ALIGN="right">
```

```
</TD></TR></TABLE>
<TABLE CELLSPACING="2" CELLPADDING="0" BORDER="0" WIDTH="100%"><TR><TD
ALIGN="left" VALIGN="top"><P CLASS="code_header">function
Product::GetParents()</P></TD></TR></TABLE>
<P>
```

Gets the parent dependencies of the product

```

<P>
<DL>
<DT CLASS="attribute">Returns
<DD CLASS="attribute_item">
  LinkedList of Product.
</DD></DT>
</DL>
<P>
<P>
<P>
<HR CLASS="divider">
<TABLE WIDTH="100%" BORDER="0"><TR><TD>
<H3 CLASS="routine"><A NAME="_Product::GetChildren">
Product::GetChildren
</A></H3>
</TD><TD ALIGN="right">

</TD></TR></TABLE>
<TABLE CELLSPACING="2" CELLPADDING="0" BORDER="0" WIDTH="100%"><TR><TD
ALIGN="left" VALIGN="top"><P CLASS="code_header">function
Product::GetChildren()</P></TD></TR></TABLE>
<P>

  Gets the children referencies.

<P>
<DL>
<DT CLASS="attribute">Returns
<DD CLASS="attribute_item">
  LinkedList of Products
</DD></DT>
</DL>
<P>
<P>
<P>
<HR CLASS="divider">
<TABLE WIDTH="100%" BORDER="0"><TR><TD>
<H3 CLASS="routine"><A NAME="_Product::MakeProduct">
Product::MakeProduct
</A></H3>
</TD><TD ALIGN="right">

</TD></TR></TABLE>
<TABLE CELLSPACING="2" CELLPADDING="0" BORDER="0" WIDTH="100%"><TR><TD
ALIGN="left" VALIGN="top"><P CLASS="code_header">procedure
Product::MakeProduct</P></TD></TR></TABLE>
<P>

```

Executes the plugin associated with the Product to generate the output data.

<P>

<P>

<P>

<HR CLASS="divider">

<TABLE WIDTH="100%" BORDER="0"><TR><TD>

<H3 CLASS="routine">

Product::SetPlugin

</H3>

</TD><TD ALIGN="right">

</TD></TR></TABLE>

<TABLE CELLSPACING="2" CELLPADDING="0" BORDER="0" WIDTH="100%"><TR><TD ALIGN="left" VALIGN="top"><P CLASS="code_header">procedure Product::SetPlugin, plugin</P></TD ></TR></TABLE>

<P>

Sets the Plugin object on which is based the Product.

<P>

<TABLE CELLPADDING="3" CELLSPACING="0" CLASS="parameters">

<TR BGCOLOR="#EEEEFF" CLASS="small_title">

<TD COLSPAN=2>Parameters</TD>

</TR>

<TR BGCOLOR="white" CLASS="parameters">

<TD ALIGN="right" VALIGN="top" WIDTH="1%">

plugin

<NOBR>in, required</NOBR>

<NOBR>Plugin</NOBR>

</TD>

<TD VALIGN="top">

Plugin on which is

related.

</TD>

</TR>

</TABLE>

<P>

<P>

<HR CLASS="divider">

<TABLE WIDTH="100%" BORDER="0"><TR><TD>

<H3 CLASS="routine">

Product::SetPluginInfo

</H3>

</TD><TD ALIGN="right">

</TD></TR></TABLE>

<TABLE CELLSPACING="2" CELLPADDING="0" BORDER="0" WIDTH="100%"><TR><TD ALIGN="left" VALIGN="top"><P CLASS="code_header">procedure Product::SetPluginInfo, plugin_info </P></TD></TR></TABLE><P>

Sets the PluginInfo object that stores the Plugin information.

<P>

<TABLE CELLPADDING="3" CELLSPACING="0" CLASS="parameters">

<TR BGCOLOR="#EEEEFF" CLASS="small_title">

<TD COLSPAN=2>Parameters</TD>

</TR>

<TR BGCOLOR="white" CLASS="parameters">

<TD ALIGN="right" VALIGN="top" WIDTH="1%">

plugin_info

<NOBR>in, required</NOBR>

<NOBR>PluginInfo</NOBR>

</TD>

<TD VALIGN="top">

PluginInfo object.

</TD>

</TR>

</TABLE>

<P>

<P>

<HR CLASS="divider">

<TABLE WIDTH="100%" BORDER="0"><TR><TD>

<H3 CLASS="routine">

Product::AssocPluginInfo2Plugin

</H3>

</TD><TD ALIGN="right">

</TD></TR></TABLE>

<TABLE CELLSPACING="2" CELLPADDING="0" BORDER="0" WIDTH="100%"><TR><TD

ALIGN="left" VALIGN="top"><P CLASS="code_header">procedure

Product::AssocPluginInfo2Plugin</P></TD></TR></TABLE>

<P>

Creates an association from PluginInfo to Plugin object of the Product.

<P>

```

<DL>
<DT CLASS="attribute">Pre-condition
<DD CLASS="attribute_item">
  Product object must has references to a valid Plugin and
  PluginInfo objects.
</DD></DT>
</DL>
<P>
<P>
<P>
<HR CLASS="divider">
<TABLE WIDTH="100%" BORDER="0"><TR><TD>
<H3 CLASS="routine"><A NAME="_Product::GetPlugin">
Product::GetPlugin
</A></H3>
</TD><TD ALIGN="right">

</TD></TR></TABLE>
<TABLE CELLSPACING="2" CELLPADDING="0" BORDER="0" WIDTH="100%"><TR><TD
ALIGN="left" VALIGN="top"><P CLASS="code_header">function
Product::GetPlugin()</P></TD></TR></TABLE>
<P>

Gets the Plugin object on which the Product is based.

<P>
<DL>
<DT CLASS="attribute">Returns
<DD CLASS="attribute_item">
  Plugin
</DD></DT>
</DL>
<P>
<P>
<P>
<HR CLASS="divider">
<TABLE WIDTH="100%" BORDER="0"><TR><TD>
<H3 CLASS="routine"><A NAME="_Product::GetPluginInfo">
Product::GetPluginInfo
</A></H3>
</TD><TD ALIGN="right">

</TD></TR></TABLE>
<TABLE CELLSPACING="2" CELLPADDING="0" BORDER="0" WIDTH="100%"><TR><TD
ALIGN="left" VALIGN="top"><P CLASS="code_header">function
Product::GetPluginInfo()</P></TD></TR></TABLE>
<P>

```

Gets the PluginInfo object with the information of the Plugin on which the Product is based.

```
<P>
<DL>
<DT CLASS="attribute">Returns
<DD CLASS="attribute_item">
  PluginInfo
</DD></DT>
</DL>
<P>
<P>
<P>
<HR CLASS="divider">
<FONT CLASS="tagline">Produced by IDLdoc 1.5 on Tue Mar 15 15:50:42 2005</FONT>
</BODY></HTML>
```

File Attachments

1) [product__define.html](#), downloaded 144 times

Subject: Re: creating documentation file
Posted by [David Fanning](#) on Tue, 15 Mar 2005 16:28:37 GMT
[View Forum Message](#) <> [Reply to Message](#)

Antonio Santiago writes:

```
> I created my own perl script:
> http://asantiago.gentelibre.org/index.php/archives/2004/12/0
1/a-perlscript-for-idl-documentation-generator/
>
> but I change to IDLdoc when I found it:
> http://asantiago.gentelibre.org/index.php/archives/2005/03/0 1/idldoc-documentation-utility/
>
> Also attach a file that use IDLdoc syntax.
```

Thank you. I can't do any real work today -- puppies are on the way and I am up and down the stairs every five minutes or so. Documentation is something to think about. :-)

Cheers,

David

P.S. Twenty-one years ago today by oldest son was born. Surprising, the flood of memories puppies can bring. Sometimes I can barely see down there through the misty

eyes. Where in the world have all those years gone!?
It seems like yesterday, literally.

--

David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: <http://www.dfanning.com/>

Subject: Re: creating documentation file
Posted by [urkhaa](#) on Tue, 15 Mar 2005 16:55:26 GMT
[View Forum Message](#) <> [Reply to Message](#)

Thank you all for the suggestions. I will give a try both to the perl script of Antonio and to the IDLdoc utility although I found a little bit boring the use of @ character for the various tags (I also use IDLWAVE and which uses the simple

```
;&+  
; Name:  
; Description:  
;-  
and so on....
```

n

Subject: Re: creating documentation file
Posted by [urkhaa](#) on Wed, 16 Mar 2005 09:13:46 GMT
[View Forum Message](#) <> [Reply to Message](#)

urkhaa@hotmail.com wrote:

```
> Thank you all for the suggestions. I will give a try both to the perl  
> script of Antonio and to the IDLdoc utility although I found a little  
> bit boring the use of @ character for the various tags (I also use  
> IDLWAVE and which uses the simple  
> ;&+  
> ; Name:  
> ; Description:  
> ; -  
> and so on....  
>  
> n
```

Here am I again....

I have to admit that I really do not understand how to use correctly IDLdoc. Actually it seems to me that if I would like to create a single HTML file for all my library (which links to single file description) I

have to previously create an overview file which in some way link to all the routines (actually I have a directory for my library and other subdirectories inside it) ... but I really do not understand how to make it. Maybe this is a stupid question and I apologize for it if it is the case but can anyone help me?

thank you all
n

Subject: Re: creating documentation file
Posted by [David Fanning](#) on Wed, 16 Mar 2005 12:28:01 GMT
[View Forum Message](#) <> [Reply to Message](#)

urkhaa@hotmail.com writes:

- > Here am I again....
- > I have to admit that I really do not understand how to use correctly
- > IDLdoc. Actually it seems to me that if I would like to create a single
- > HTML file for all my library (which links to single file description) I
- > have to previously create an overview file which in some way link to
- > all the routines (actually I have a directory for my library and other
- > subdirectories inside it) ... but I really do not understand how to
- > make it. Maybe this is a stupid question and I apologize for it if it
- > is the case but can anyone help me?

Here is the routine I've used for years to document my Coyote Library. It's simple, efficient, and results in a single HTML page. It lacks a couple of features I would like to have (and DocLib contains), but I agree: I'm not sure DocLib is so much better it is worth reworking ALL my documentation!

```
.,*****  
,  
PRO HTML_Doc  
  
CD, Current=thisDir  
CD, 'C:\coyoteguide\programs'  
files = Findfile('*.pro')  
  
MK_HTML_Help, files, 'program.documentation.html', Title='IDL Example  
Programs'  
  
; Make the IDLWAVE Catalog file.  
  
SPAWN, 'perl C:/CoyoteGuide/Miscellaneous/idlwave_catalog $  
-f Coyote', /NoShell  
CD, thisDir
```

END

,*****
,

Cheers,

David

--

David Fanning, Ph.D.

Fanning Software Consulting, Inc.

Coyote's Guide to IDL Programming: <http://www.dfanning.com/>

Subject: Re: creating documentation file

Posted by [Antonio Santiago](#) on Wed, 16 Mar 2005 12:38:52 GMT

[View Forum Message](#) <> [Reply to Message](#)

This is an example to show how I work with IDLdoc.

Suppose my project directory is:

MyProject

```
|-- main.pro
|
|---Subdir1
| |--File1.pro
| \--File2.pro
|
|---Subdir2
| |--FileA.pro
| \--FileB.pro
|
|--Doc
  |--API_doc(subdir)
  \--idldoc (subdir)
```

I want to create thye API documentation for all *.pro files of my project and put it into API_doc subdirectory.

I have put all IDLdoc files into 'idldoc' subdirectory (idldoc.sav, idldoc.css, etc). Then I have created a file called "gendoc" (Generate documentation :)) like this:

PRO gendoc

```
idldoc, root='.././', $
      output='../api', $
      title='GenRad 2.0', $
      subtitle='GRAHI', $
      ;footer='../foot.html'
```

END

The root directory of my project is "two levels up" (root=../..).
I want to put the HTML generated files into API_doc subdirectory (in
this case: output=../API_doc).
Also I want my own title and subtitle :)

The only problem I have found is with "footer" keyword. I don't know if
I have created a bad foot.html but if I include this option the
resulting overview.html file is not correct :(

Well, to generate the documentation just go into de MyProject/Doc/idldoc
directory and execute "gendoc".

Bye,
Antonio.
