Doc No: UG60917-IDP-036
Rev -
Date: 04 January 2012
Contract No. NNG10XA03C

JOINT POLAR SATELLITE SYSTEM (JPSS) COMMON GROUND SYSTEM (CGS)

IDPS PRO XML EDITOR USERS MANUAL

CDRL No. A032

RAYTHEON COMPANY
INTELLIGENCE AND INFORMATION SYSTEMS (IIS)
JPSS CGS PROGRAM
AURORA, COLORADO

IAW DFARS 252.227-7036, Raytheon hereby declares that, to the best of its knowledge and belief, the technical data delivered under Contract No. NNG10XA03C is complete, accurate, and complies with all requirements of the Contract.

TITLE: JOINT POLAR SATELLITE SYSTEM (JPSS) COMMON GROUND SYSTEM (CGS) IDPS PRO XML EDITOR USER'S MANUAL

APPROVAL SIGNATURES:
(Signatures indicate compliance with these requirements)

William Sullivan

Digitally signed by William Sullivan
DN: cn=William Sullivan, o=JPSS, ou=DCMS, email=wjsullivan@raytheon.com, c=US
Date: 2012.01.12 10:47:24 -07'00'

William Sullivan Date
Program Manager

Frank Zorn

Digitally signed by Frank Zorn
DN: cn=Frank Zorn, o=JPSS Sustainment, ou=JPSS Sustainment, email=fgzorn@raytheon.com, c=US
Date: 2012.01.06 13:54:34 -07'00'

Frank Zorn Date
Sustainment IPT Lead

Harvey Lee

Digitally signed by Harvey Lee
DN: cn=Harvey Lee, o=JPSS, ou=JPSS, email=hylee@raytheon.com, c=US
Date: 2012.01.10 09:44:01 -07'00'

Harvey Lee Date
IDPS Sustainment Manager

Justin Lee

Digitally signed by Justin Lee
DN: cn=Justin Lee, o=IDPS Sustainment, ou=JPSS, email=justin.lee@raytheon.com, c=US
Date: 2012.01.10 09:49:55 -07'00'

Justin Lee Date
Software Manager

Sheryl Fertman

Digitally signed by Sheryl Fertman
DN: cn=Sheryl Fertman, o=IIS QA, ou=IIS QA, email=slfertman@raytheon.com, c=US
Date: 2012.01.10 13:17:43 -07'00'

Sheryl Fertman Date
Quality

Revision/Change History

Revision	Document Date	Revision/Change Description	Pages Affected
-	04 Jan 2012	Initial release per ECR-IDP-004061	All

Table of Contents

List of Figures

List of Tables

1 SCOPE

The PRO XML Editor is a tool used to develop and maintain the PRO product XML format files.

1.1 DOCUMENT OVERVIEW

This document provides information to understand the PRO XML Editor functionality, specifically how the tool works, inputs necessary to execute the tool, outputs that the tool generates, the Human Machine Interface (HMI), and tool limitations.

The person using this tool is generally referred to as the User.

2 REFERENCE DOCUMENTS

The documents listed in Table 2-1 are applicable to this specification and the content herein.

TABLE 2-1 REFERENCED DOCUMENTS

Document Number	Title
UG60917-IDP-034	IDPS PRO SW User Manual Part 1
UG60917-IDP-034	IDPS PRO SW User Manual Part 2
LI60917-GND-005	JPSS CGS Acronyms & Glossary
474-00001-01 through -08	JPSS Common Data Format Control Book – External, Vol I – Vol VIII

3 OVERVIEW

The PRO XML Editor operations are governed by the PRO XML Editor operator's actions. The operator invokes the PRO XML Editor, which results in the tool's HMI being displayed.

The PRO XML Editor HMI allows users to create, edit, validate and save PRO XML. The PRO XML Editor also allows the user to create PRO XML from CDFCB XML and to save from PRO XML into CDFCB XML format.

3.1 SUPPORTED PLATFORMS

The PRO XML Editor is currently supported on IBM computers running the IBM AIX operating system and on platforms running Red Hat Enterprise Linux 5 for ADL. The resource requirements, in terms of memory, storage, computing cycles, etc, are currently unavailable.

4 ENVIRONMENT SETUP

4.1 PREPARATION

The PRO XML Editor requires Java 1.6 64, ant, and log4j. The environment variable JAVA_HOME must point to the Java 1.6 64 bit install directory and $JAVA_HOME/bin must be in your PATH variable. Table 4-1 shows the needed environment variables that must be defined to compile and run the editor.

TABLE 4-1 COMPILE-TIME ENVIRONMENT VARIABLES

Variable Name	Value
HOME	User home directory path
JAVA_HOME	Path to Java 1.6 64 bit version
ANT_HOME	Path to the Apache Ant build utility's home directory
LOG4J_JAR_FILE	Path to log4j JAR file
XML_EDITOR_BASEDIR	The base directory of the XML editor's source and configuration files. This variable is necessary for disambiguating the directory structures of the IDPS processing and ADL development environments.
PRO_XML_SCHEMA	Path to the PRO XML schema; correct setting of this variable depends on the correct setting of the XML_EDITOR_BASEDIR variable
DFCB_XML_SCHEMA	Path to the CDFCB XML schema; correct setting of this variable depends on the correct setting of the XML_EDITOR_BASEDIR variable
PATH	User's PATH variable must contain $JAVA_HOME/bin so that the ant process is able to execute the xjc schema compiler.

4.2 RUNTIME ENVIRONMENT VARIABLES

The run time environment also depends on the correct setting of the environment variables listed in Table 4-1. Also, tool tips are supported in the tool. However, the path to the input text file containing the tool tips must be specified via an environment variable "XML_HELP". The name of the input file provided is called xml_help.txt, located in the <INSTALL_DIR>/cfg directory.

TABLE 4-2 RUNTIME ENVIRONMENT VARIABLES

Variable Name	Value
XML_HELP	Path to the xml_help.txt input file

4.3 COMPILING THE XML EDITOR

To compile the XML Editor, the environment variables in Table 4-1 must be set properly. Then go to <INSTALL_DIR>/java and execute ant. This will compile the tool with the output jars in a subdirectory called dist. Other build targets include javadocs (to build the javadoc) and clean,

to remove all compiled byte code, the JAR file, javadocs, log file (created after running the tool), and directories created by the original ant build.

4.4 STARTING PRO XML EDITOR

To run the XML Editor, execute one of the following scripts: 1) for ADL users, ${ADL_HOME}/script/xmled, 2) for development users, ${PRO_HOME}/script/xmled, 3) for ADA users, ${DPE_ROOT_PATH}/cat3/script/runAdaXmlGui.csh. Figure 4.4-1 shows the starting edit window displayed on start up. The PRO XML Editor allows the users to modify all entries in PRO XML with the exception of count variables. The editing feature works by allowing users to change sections of XML by invoking dialog wizards that prompt the user for all needed inputs. Any missing inputs are highlighted red to notify the user to complete filling out the needed information. Some wizards will enable or disable fields automatically because they depend on values the user has set in other fields. For instance, there may be a drop down menu where either true or false is possible and a text area with free text input that will only be enabled if the value in the previous menu is true. Figure 4.4-2 shows an open XML file.

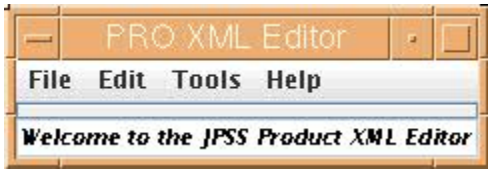

FIGURE 4.4-1 PRO XML EDITOR MAIN WINDOW

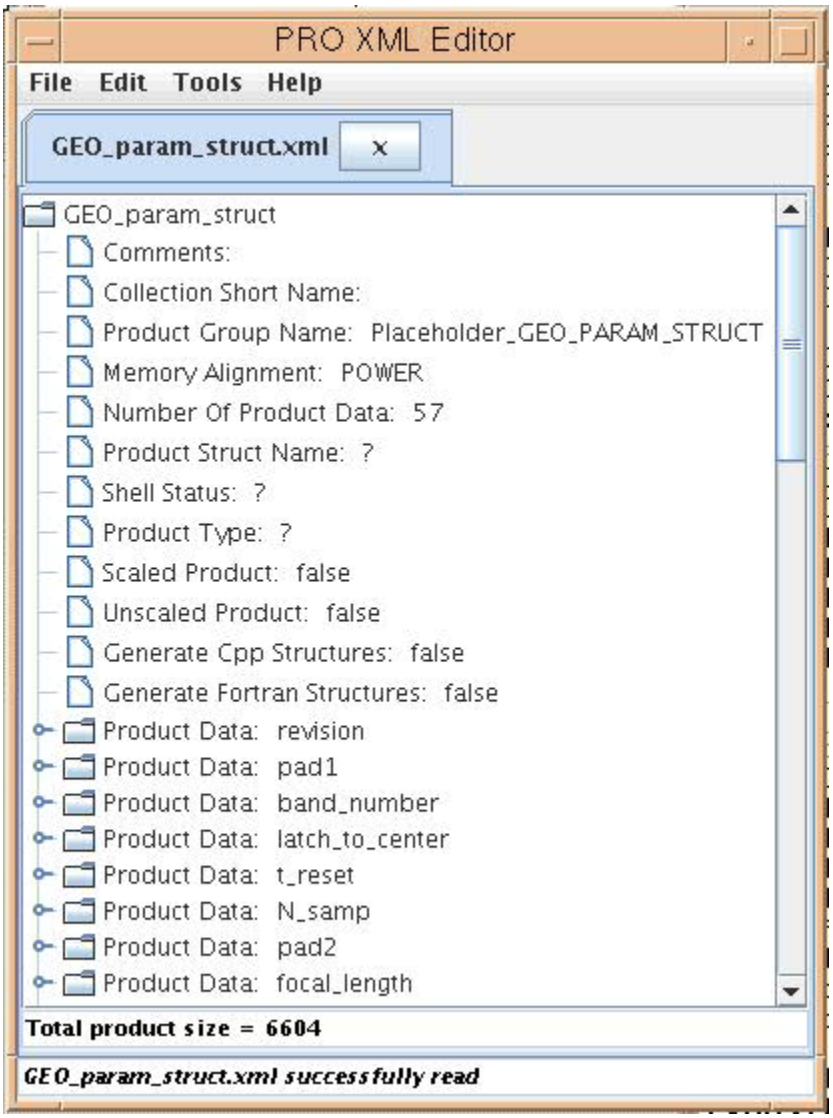

FIGURE 4.4-2 XML EDITOR MAIN DISPLAY

Editing is initiated by right clicking an element and selecting, one of the three options, edit, insert, or remove. Edit will open the dialog wizard needed to update that selected element. Remove will remove the element if allowed by the schema. Insert will allow the user to insert a new element of the selected type following the selected element if allowed. Figure 4.4-3 displays the options windows on a right click.

FIGURE 4.4-3 EDIT OPTIONS

Editing can also be initiated by double clicking an element. The edit dialog for that element is opened in the same manner as right clicking and selecting the edit option. Once a Product Data node is inserted, it can be moved above or below other Product Data nodes by a "drag-and-drop" action. Note that drag-and-drop capability only exists for Product Data nodes.

4.4.1 Product Level Edit

The product level can only be edited because it is the top level item. By right clicking on any of the top level items and selecting "Edit Entry", the edit product dialog wizard is displayed as shown in Figure 4.4-4. Each entry can be modified as desired. The items listed in Table 4-3 must be completed in order to close the dialog wizard and update the fields.

TABLE 4-3 PRODUCT DIALOG WIZARD REQUIRED FIELDS

Required Fields
Product Data

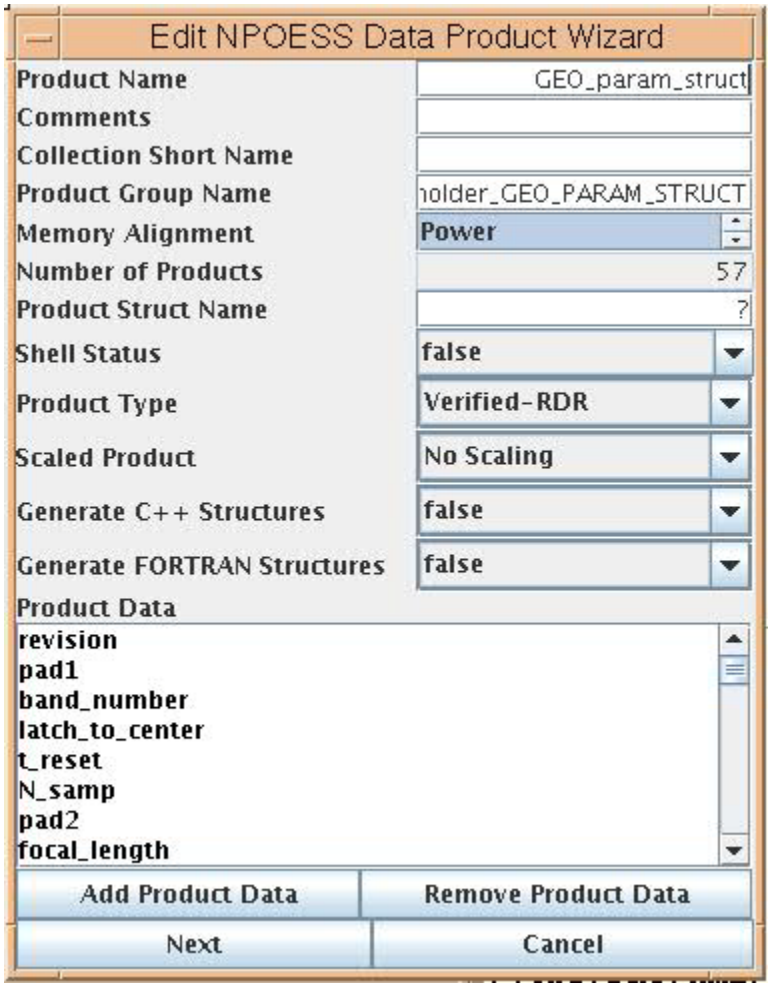

FIGURE 4.4-4 EDIT PRODUCT DIALOG WIZARD

4.4.1.1 Product Name

The product name is the name of the overall product being defined with this XML. The product name is generally the same as the collection short name as described in the CDFCB. This is done for ease of use, and does not have to match.

4.4.1.2 Comments

These are programmatic comments for the owning data product.

4.4.1.3 Collection Short Name

This is the name of the product as described in all official documentation. The short name is used by all external users of the product for describing a given product.

4.4.1.4 Product Group Name

The product group name is used by the algorithms to reference the algorithm's configuration guide to look up a given product's short name. This abstracts the short name away from the code. The product group name is part of the key in finding a product definition in the software that is created from parsing the XML.

4.4.1.5 Memory Alignment

This field defines the data alignment used for the product. There are three options: Natural, Power, or Packed. Natural is the default selection when defining new products and will force data fields to be placed on natural word boundaries. Power should be selected when it's desired to have data fields placed on IBM Power word boundaries. The Editor will automatically add implicit pad fields of the appropriate size when misaligned data fields exist under either of these memory alignment schemes. Packed should be selected when it's desired to have a packed product (i.e. no data alignment).

4.4.1.6 Number of Products

This field is automatically populated based on the number of product data entries in this file.

4.4.1.7 Product Struct Name

This is the name of the C++ structure that is in the algorithm software code that this XML describes. If the Generate C++ Structure flag (See 4.4.1.11) is on, code can be auto generated with this name as the structure symbol name.

4.4.1.8 Shell Status

This field identifies products that can be produced as shell products from the algorithm. Shell products are just metadata entries in DMS to show that the algorithm was run, but no data was generated. This currently applies to night granules in the VIIRS chain only for algorithms which only produce output during the day.

4.4.1.9 Product Type

The product type is the type of record this xml represents. The main types of records are RDR, SDR, EDR, IP, and AUX. Each type of data is described in the CDFCB.

4.4.1.10 Scaled Product

This item has three options: no scaling, scaled, or un-scaled. No scaling should be selected if the product described in the xml is neither a scaled product nor an unscaled product (this applies to most products). Scaled product should be selected if the product described in the xml is a scaled version of a product and the PRO/ADL Dictionary software is to do the scaling. Un-scaled should be selected if the product described in the xml is an un-scaled version of a product and the PRO/ADL Dictionary software is to do the un-scaling.

4.4.1.11 Generate C++ Structure

The Generate C++ Structure flag tells the PRO and ADL build software to generate C++ structures from this XML for use in algorithm development.

4.4.1.12 Generate FORTRAN Structures

The Generate FORTRAN Structures flag tells the PRO and ADL build software to generate FORTRAN structures from this XML for use in algorithm development.

4.4.1.13 Product Data

The product data list shows all the product data describing the contents of this product.

4.4.2 Product Data Level Edit

The Product Data Edit Dialog Wizard as shown in Figure 4.4-5 can be opened by right clicking an existing Product Data element and selecting “Insert” or “Edit”. “Insert” will create a new Product Data element following the selected element. Table 4-4 shows the required fields that must be completed to update the entries.

TABLE 4-4 PRODUCT DATA DIALOG WIZARD REQUIRED FIELDS

Required Fields
Product Data Name
Fields

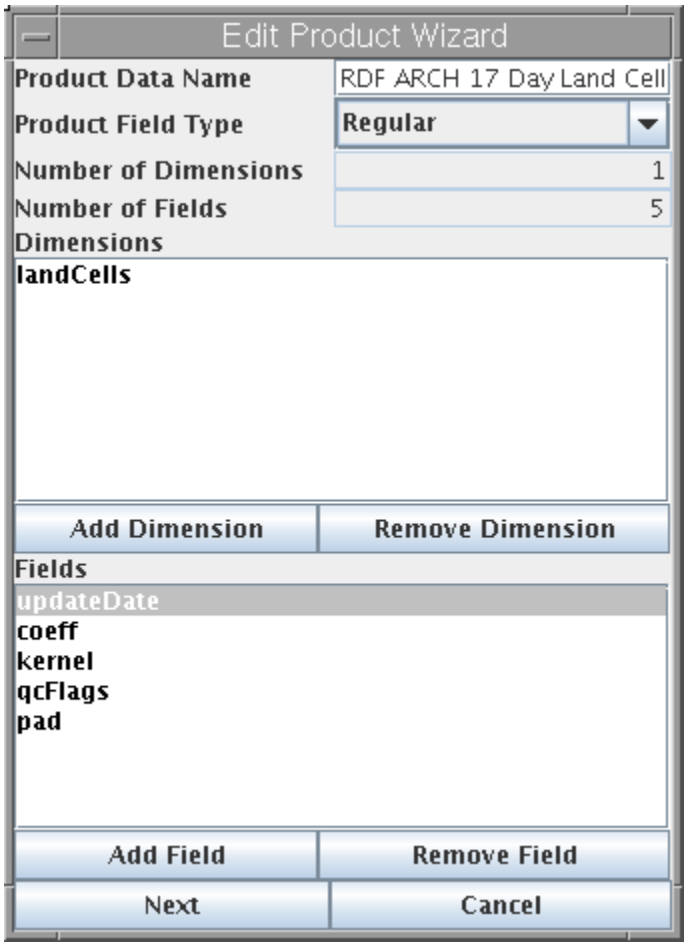

FIGURE 4.4-5 EDIT PRODUCT DATA DIALOG WIZARD

4.4.2.1 Product Data Name

Product data is a logical sub-section of a given product. Each Product Data can have one or more Fields to describe the data being stored. The Product Data Name is the logical name given to the fields of this section.

4.4.2.2 Product Field Type

The Product Field type describes the type of data in this logical sub-section of the product.

Quality defines quality flags. Pad is used to describe pad areas in the product. ScaleFactor is used to describe Scale/Offset data for a scaled product. Regular covers all other types of data.

4.4.2.3 Number of Dimensions

The number of Dimensions is the count of dimensions for the fields in this product data and is automatically populated.

4.4.2.4 Number of Fields

The number of Fields is the count of fields for this product data and is automatically populated.

4.4.2.5 Fields

Fields are the data elements for this product data.

4.4.3 Field Edit

The Field Edit Dialog Wizard, as shown in Figure 4.4-6, can be opened by right clicking an existing Field element and selecting insert or edit. Insert will create a new Field element following the selected element. The table below shows the required fields that must be completed to update an entry.

TABLE 4-5 FIELD DIALOG WIZARD REQUIRED FIELDS

Required Fields
Field Name
Datum

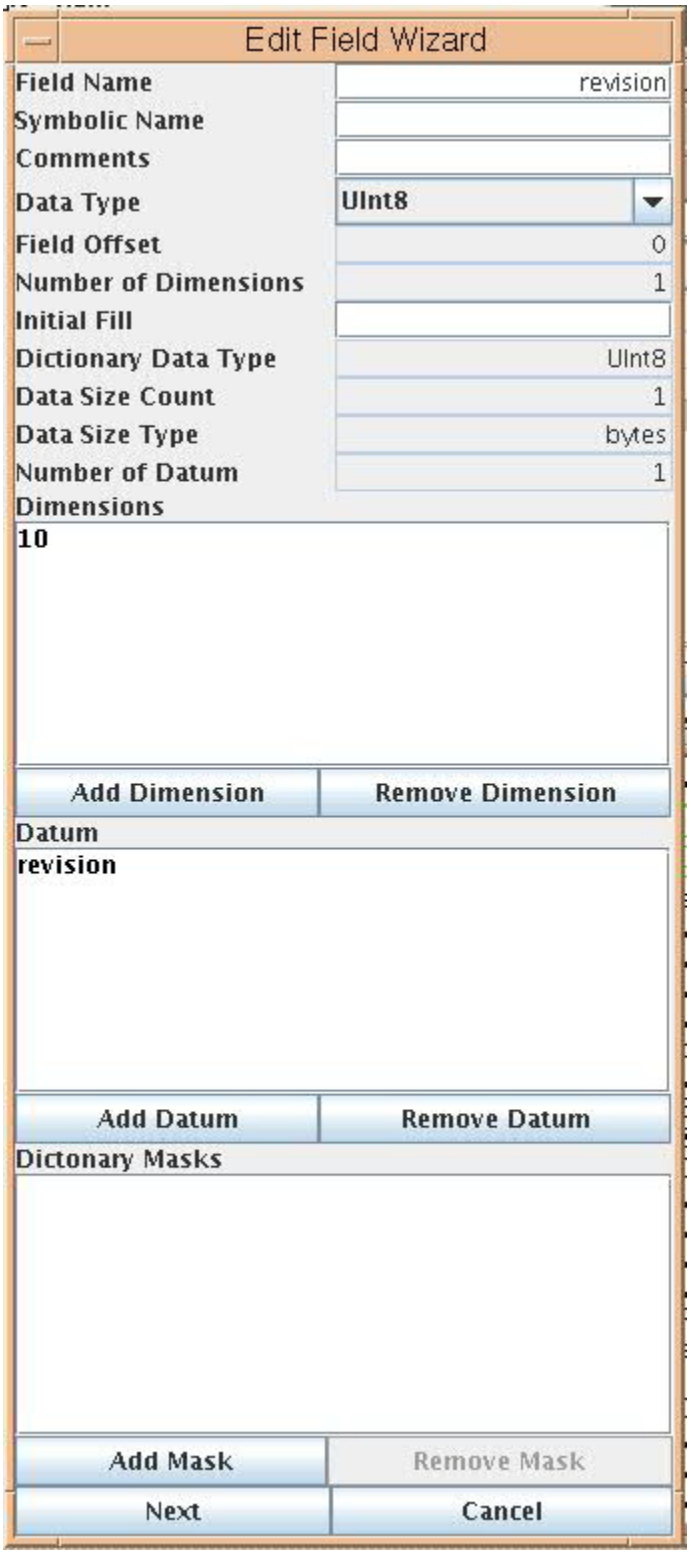

FIGURE 4.4-6 EDIT FIELD DIALOG WIZARD

4.4.3.1 Dimensions

These are product level Dimensions that used primarily for dynamic GRIDIP products. Product Data Dimensions are a feature that is not yet available in the current version of ADL.

4.4.4 Dimension Edit

The Dimension Edit Dialog Wizard as shown in Figure 4.4-8 can be opened by right clicking an existing Dimension element and selecting insert or edit. Insert will create a new Dimension element following the selected element. Table 4-7 shows the required fields that must be completed to update the entries. Alternatively, an existing dimension may be selected from already completed XML definitions. The editor must be told in which directory the XML resides, however. For instructions on how to do so, refer to Section 4.5.8.

4.4.4.1 Field Name

This is the logical name of the field being described.

4.4.4.2 Symbolic Name

The symbolic name of the field as it would appear in the C or FORTRAN structure definition. This field is optional, because a name can be derived from the logical field name, as long as the logical field name has no other special characters than spaces or underscores. If the logical field does have special characters other than spaces or underscores (or it begins with a number), then this field should be used. If this field doesn't follow C or FORTRAN coding standards then the auto generated software will fail to compile. If this field is not used and a symbolic name cannot be successfully derived from the logical field name, then the auto generated software will fail to compile. As previously noted in Sections 4.4.1.11 and 4.4.1.12, code is auto generated if the Boolean values described in the respective sections are set to true.

4.4.4.3 Comments

Programmatic comments for the owning field.

4.4.4.4 Field Offset

Offset of the field in the product data. This field is automatically populated.

4.4.4.5 Number of Dimensions

Denotes the number of array dimensions in this field. This field is automatically populated.

4.4.4.6 Initial Fill

The initial fill value is used to populate memory when the structure is created in the algorithm. This allows you to define what the default value for each entry for this field is.

The data must be within valid range for this field, and be of the correct data type.

4.4.4.7 Data Type

This selection box allows you to select what type of data the field is. This will automatically change the values in the next four fields. If this field is a quality or scale factors product field type (refer to Section 4.4.2.2) then this option box will automatically be set to UInt8 or Float32, respectively. There are few, if any situations, where a quality flag field is anything other than a single byte per item. If an exception is ever encountered, however, then change the product field type to regular to allow more flexibility.

This type is restricted to valid types of data for the PRO algorithms.

4.4.4.8 Dictionary Data Type

The type of data the PRO Data Dictionary will use for conversions. This value will always match the data type unless it is a scale/offset value and is automatically populated.

When the field is a scale/offset value, the Dictionary type is automatically updated to the correct type for a scale/offset value.

4.4.4.9 Data Size Count

Data Size count represents the size of the Data Type above in bytes. This value is automatically updated.

4.4.4.10 Data Size Type

Data Size Type will always be byte.

4.4.4.11 Number of Datum

Number of Datum is the count of the Datum descriptions for this field. This value is automatically populated.

4.4.4.12 Dictionary masks

The dictionary masks are a set of rules for how the field is handled by the PRO/ADL dictionary software. Figure 4.4-7 shows the dialog for adding new entries. Table 4-6 explains the entries.

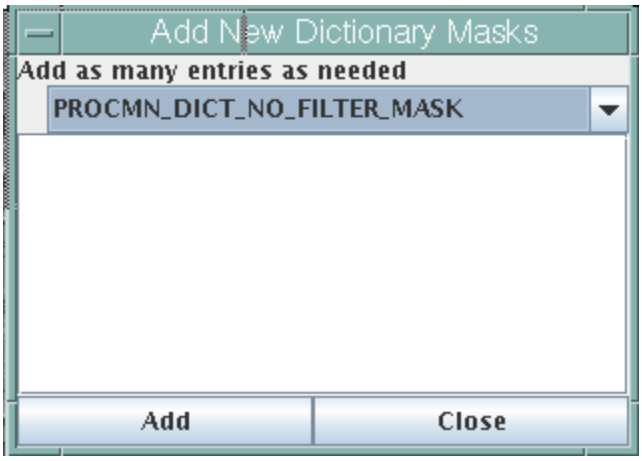

FIGURE 4.4-7 ADD DICTIONARY MASK DIALOG

TABLE 4-6 DICTIONARY MASK DESCRIPTIONS

Fill Value	Purpose
PROCMN_DICT_NO_FILTER_MASK	No filtering of data is done
PROCMN_DICT_DEGTORAD_MASK	The data for this field can be converted from degrees to radians.
PROCMN_DICT_SCALE_F_32_TOI_16_MASK	Scale the data from Float32 into Int16
PROCMN_DICT_SCALE_F_32_TOI_08_MASK	Scale the data from Float32 into Int8
PROCMN_DICT_NOCOPY_FILL_MASK	Do not copy the data, set for scale/offset fields in a scaled product to denote that these fields are not copied when trying to convert a scaled product to the unscaled version
PROCMN_DICT_MOD_FILL_MASK	Fill the data with VIIRS Moderate Band filling accounting for on board, on ground and data does not exist fill values
PROCMN_DICT_IMG_FILL_MASK	Fill the data with VIIRS Imagery Band filling accounting for on board, on ground and data does not exist fill values
PROCMN_DICT_SCAN_FILL_MASK	Fill the last scan of data with data does not exist fill in case this granule is a short granule
PROCMN_DICT_FILL_TEST_MASK	This mask is for populating the N_Percent_Missing/NA/Erroneous Operational metadata, and only SDR/TDR/EDR/IP floating point fields should use it
PROCMN_DICT_FATSCANFIELD_MASK	This indicates that this scan field needs to be extended, if extended granule is on
PROCMN_DICT_FATGRANULEFIELD_MASK	This indicates that this granule field needs to be extended, if extended granule is on.

4.4.4.13 Dimensions

The dimensions will represent the field layout in memory for an array of data. Each entry is an index into the array of data.

4.4.4.14 Datum

The datum describes the layout of a single entry in the field. The total count and size of the datum must be equal to the size of the data.

4.4.5 Dimension Edit

The Dimension Edit Dialog Wizard as shown in Figure 4.4-8 can be opened by right clicking an existing Dimension element and selecting insert or edit. Insert will create a new Dimension element following the selected element. Table 4-7 shows the required fields that must be completed to update the entries. Alternatively, an existing dimension may be selected from already completed XML definitions. The editor must be told which directory these XML reside, however. For instructions on how to do so, refer to Section 4.5.8.

TABLE 4-7 DIMENSION DIALOG WIZARD REQUIRED FIELDS

Required Fields
Dimension Name
Granule Boundary
Dynamic
Minimum Index
Maximum Index

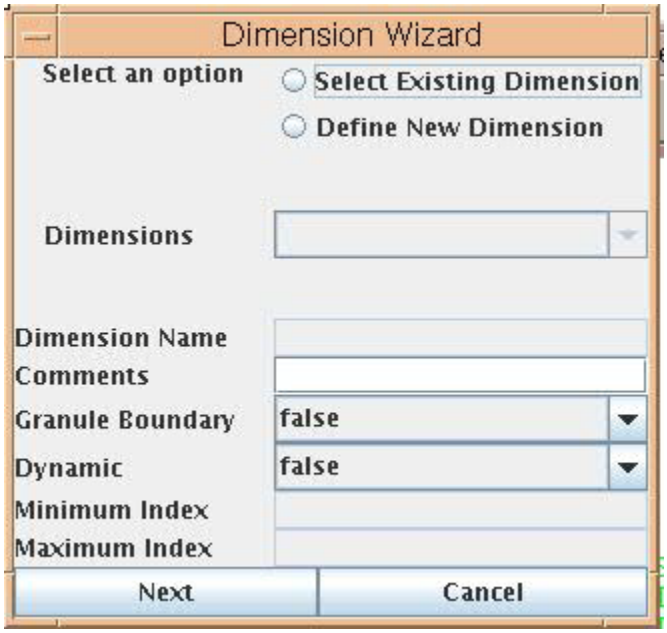

FIGURE 4.4-8 EDIT DIMENSION DIALOG WIZARD

4.4.5.1 Dimension Name

Logical name for this dimension, these names get parsed into auto generated constants for use in C++ structure definitions if Generate C++ Structures flag is enabled.

4.4.5.2 Comments

Programmatic comments for the owning dimension.

4.4.5.3 Granule Boundary

The Granule Boundary element indicates that the dimension is contiguous over granule boundaries. For scanning sensors, i.e. VIIRS, the cross-track dimension is the dimension that lies along granule boundaries. That is, the first value of the cross-track dimension of the last in-track index of a VIIRS array is adjacent to the first value of the cross-track dimension of the first in-track index of the following granule of a VIIRS product granule array.

4.4.5.4 Dynamic

This field describes the state of the dimension to let users know if it can change dynamically based on the size of the granules being used to generate the product.

4.4.5.5 Minimum Index

The minimum index describes the smallest size of the dimension. This field should always match the maximum Index, unless the dynamic flag was marked above. If the dynamic flag is false, the wizard will disable this field and show it as empty. If the user finishes the wizard by clicking next and dynamic is still false and a valid value is entered for the maximum index, then the minimum index will automatically be populated with that value as well. If the user finishes with the dynamic flag set to true, then the minimum index must be some value less than or equal to the maximum index and greater than or equal to zero.

4.4.5.6 Maximum Index

The maximum index describes the maximum size of the dimension.

4.4.6 Datum Edit

The Datum Edit Dialog Wizard as shown in Figure 4.4-9 can be opened by right clicking an existing Datum element and selecting insert or edit. Insert will create a new Datum element following the selected element. Table 4-8 describes the required fields that must be completed to update the entries.

TABLE 4-8 DATUM DIALOG WIZARD REQUIRED FIELDS

Required Fields
Datum Name
Offset
Scaled Item
Scale Factors Node Name
Measurement Unit

Range Min
Range Max
Data Range

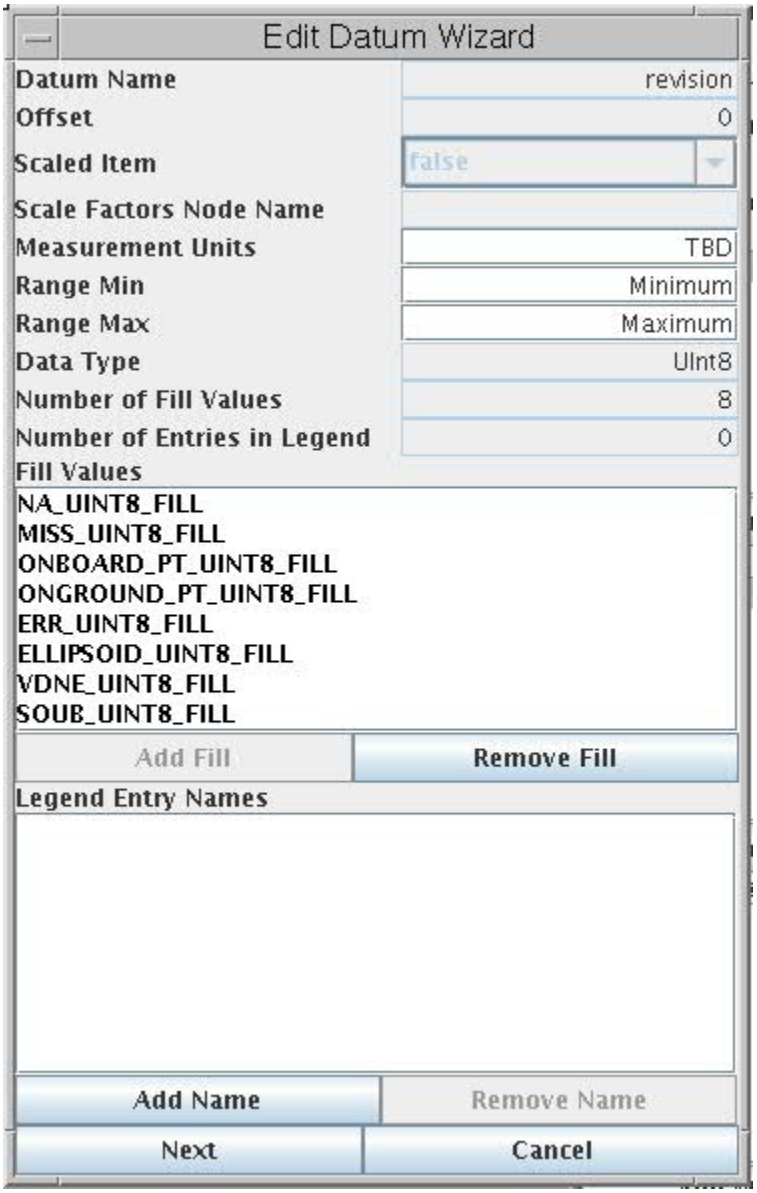

FIGURE 4.4-9 EDIT DATUM DIALOG WIZARD

4.4.6.1 Datum Name

The datum name is only editable if this is a quality product field type (refer to Section 4.4.2.2). Otherwise, the wizard automatically reuses the logical field name (refer to Section 4.4.4.1) and disables the field. This field is an ASCII description of the datum.

4.4.6.2 Offset

Datum offset in the field it is describing, this value is automatically populated.

4.4.6.3 Scaled Item

This is a Boolean value where true indicates that this is a scaled field. Note that this is only possible if the product itself is a scaled product (refer to Section 4.4.1.10), so this wizard disables this item if the product is either not scaled or unscaled.

4.4.6.4 Scale Factors Node Name

If the scaled item flag described in the previous section is true, this field is editable and must refer to the product data node that contain the scale factors for this scaled field. This field is not editable, otherwise.

4.4.6.5 Measurement Units

This field is the plain ASCII representation of the measurement units of the data stored in this datum.

4.4.6.6 Range Min

The Minimum value stored in this datum, which must be smaller than the max range.

4.4.6.7 Range Max

The Maximum value stored in this datum, which must be larger than the min range.

4.4.6.8 Data Type

The type of data this datum will describe. The type must be smaller or equal to the type of the field it describes.

This size can also be described in bits as needed, if this is a quality product field type

4.4.6.9 Number of Fill Values

Number of fill values defined for this datum. This number is automatically populated.

4.4.6.10 Fill Values

Fill values (as defined in the CDFCB-X) that apply to this field or datum. All fill values are automatically populated in this list and may be removed or reapplied as desired.

4.4.6.11 Number of Entries in Legend

This is the number of entries in the legend for this datum. This number is automatically populated.

4.4.6.12 Legend Entry Names

The legend of known values mapped into the logical name for that value. This is mostly used to define quality flag values, but is not restricted to that use. Figure 4.4-10 shows the dialog for adding new legend entries. The dialog requires the name for the entry and the value that the name represents.

FIGURE 4.4-10 ADD NEW LEGEND ENTRY DIALOG

4.5 PRO XML EDITOR MENUS

4.5.1 New

The New option will create a temporary file and start the XML Product edit window, allowing the user to create new PRO XML.

4.5.2 Open

Open a new PRO XML file. The Open option allows the user to open existing XML files that are either in PRO XML or CDFCB XML format.

4.5.3 Save

Save the current tab data using the current name to the file system.

4.5.4 Save As

Save the current tab with a new name to the file system.

4.5.5 Exit

Closes the tool, but prompts user if any changes have not been saved.

4.5.6 Preview Struct Definition

This menu item will display a report of how the structure definition would be auto-generated (in C format). This is a tool provided for convenience, so that users may ensure that the product is being developed as envisioned by the development team.

4.5.7 Detect Conflicting Dimensions

By clicking this menu item, a thread is opened that loops over all XML files in the paths defined in Section 4.5.8 and validates that there are no dimensions with conflicting definitions. If any conflicts are found by the thread, it opens a dialog containing a drop-down list of all dimension names that have conflicting definitions. Users are able to select a dimension they wish to inspect and then click the Inspect Conflicts button. A separate dialog opens displaying the XML content of the conflicts.

4.5.8 Preferences

This dialog is a placeholder for future development. It does, however, serve one useful function currently. It allows the user to tell the editor where all of the existing XML definitions currently reside. For PRO developers, these are the directories in ${PRO_HOME}/xml, and for ADL developers, ${ADL_HOME}/xml. Be warned, however, that subdirectories will not be searched, so the user must select each subdirectory individually, if they exist. The purpose of this is to pre-populate the wizards with data that has already been defined and can be reused. Currently, the only item supported are dimensions. So, if one or more directories are properly selected, then the dimension wizard should have a list of predefined dimensions that may be reused. The operator is warned, however, that changing this setting will not take effect until the next session. Also, if any errors occur while pre-populating the data, they occur silently. If you've selected valid directories and the dimension list is empty in the dimension wizard after opening a new session, check the log file for errors.

4.6 PRO XML EDITOR VALIDATION

PRO XML must follow the PRO XML Schema defined and used by the PRO XML Editor. Any XML created with the PRO XML Editor will conform to the schema.

4.6.1 Detected Errors

Any errors in the XML where the values do not conform to the XML schema are displayed in red. When errors are detected, the root elements all the way to the root element are displayed red so that the user can locate the element that does not meet the PRO XML schema.

4.6.2 Validation Updates

XML Validation is preformed in two cases. The first case is when XML is opened. The second is when XML is edited.

4.6.3 Schema Versioning

Since ADL Phase 3.0, schema versioning is enforced. A new attribute called "schemaVersion" has been inserted into each XML file's "NPOESSDataProduct" element. Also, a new attribute has been added to the XML schema's "schema" element called "version." The string values

for “schemaVersion” in the XML and “version” in the schema must match or the application will fail to open the XML file. The XML Editor may fail to open your XML due to schema version conflicts for one of three reasons: 1) the XML file does not have a “schemaVersion” attribute; 2) the schema does not have a “version” attribute; or 3) the “schemaVersion” attribute in the XML does not match the “version” attribute in the schema. Case 2 probably means that you’re using an old version of the XML editor (prior to its inception of schema version checking.) If that’s the case, then all of your software (XML and other code) should be from the same build of IDPS (or ADL) that the XML Editor software came from. If it’s not, then there’s no guarantee that it will be able to process or generate XML appropriate for that build of software. For cases 1 and 3, either the version of the XML that conforms to the schema used by the XML editor must be located, or the XML must be updated to conform to the latest version of the schema.

In order to update your XML, you must first know how the version of the schema that was used to create the XML differs from the version of the schema that you are currently attempting to use. As an example, say you are attempting to open XML developed with the XML Editor packaged with ADL Phase 3.0 and you are using the XML Editor packaged with a future version of ADL. Copies of all previous versions of the schema since the induction of schema versioning support can be found in the ${ADL_HOME}/cfg directory. The most recent version is always named pro_xml.xsd and is an exact copy of the latest. Therefore, the schema used in ADL Phase 3 is pro_xml_v1.xsd and pro_xml.xsd is a copy of it. Now, let the version used by the future version of ADL in our example be pro_xml_v2.xml. In this example, you would have to diff pro_xml_v1.xsd versus pro_xml_v2.xsd. Then, the XML would have to be updated according to what has changed since pro_xml_v1.xsd. In our example, say the diff reveals that there has since been added a new required field in the “NPOESSDataProduct” nodes called “BornOn” and it is a string. Furthermore, the field new element must fall directly after the “StructName” element. Now, the XML will have to be updated with an external application ─ such as a text editor ─ so that there is a “BornOn” element after “StructName” with an appropriate string value within the element. For the most part, elements within the PRO Product XML files are sequence-dependent (as indicated by the schema element “xs:sequence”).

4.7 PRO XML EDITOR LOGGING

The PRO XML Editor produces log information about the current run in the same directory where the user started the program. It is important that the user has write permissions for this directory. The log information provides error messages to the user if there are problems with the tool.

5 EXAMPLE WALK THOUGH

The following walkthrough of creating an XML from scratch covers the basic values that must be entered into the wizards to create XML from scratch.

As each new entry is made in the XML, the wizards populate default values into each field to help with XML creation.

This walkthrough starts with the assumption the tool is open and ready for use.

5.1 PRODUCT BEING CREATED

The product being created is going to be an EDR product with two fields.

The first field is a float data array with one dimension of 10 entries.

The second field is a quality flag with one dimension of size 10, with two legend entries explaining the quality flag data.

5.2 PRODUCT WIZARD POPULATION

Select File, New. The Product data Wizard opens and is described in section 4.4.1. Figure 5.2-1 shows the initial state of the dialog.

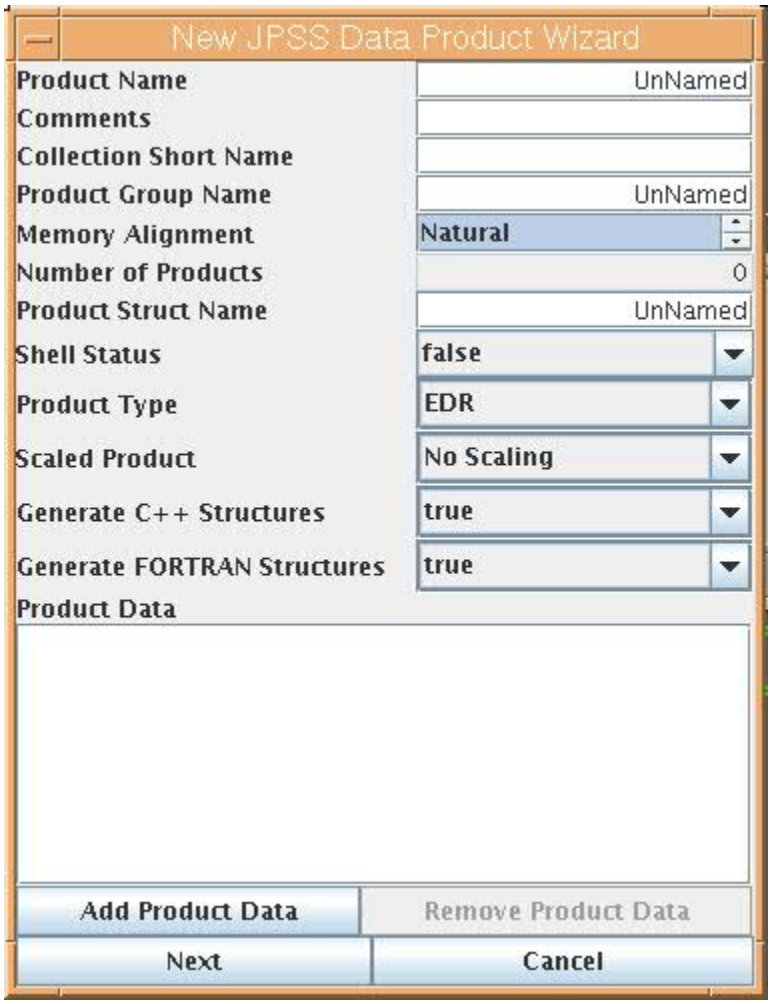

FIGURE 5.2-1 INITIAL DATA PRODUCT WIZARD DIALOG

The first step is to enter a name for the product. User will enter “Simple EDR Product” into the Product Name. User will enter “Simple Product Comment” into the Comments field. User will enter “Simple-EDR” into the Collection Short Name and enter “Simple_EDR” into the group name. The user will define the structure name as “SimpleEdrType”. We will not allow shell status and generate both C++ and FORTRAN structures.

For this example, it will not be scaled and therefore the Scaled Product flag is No Scaling, because neither scaling nor un-scaling apply to a non-scaled product.

Figure 5.2-2 shows the completed Entries.

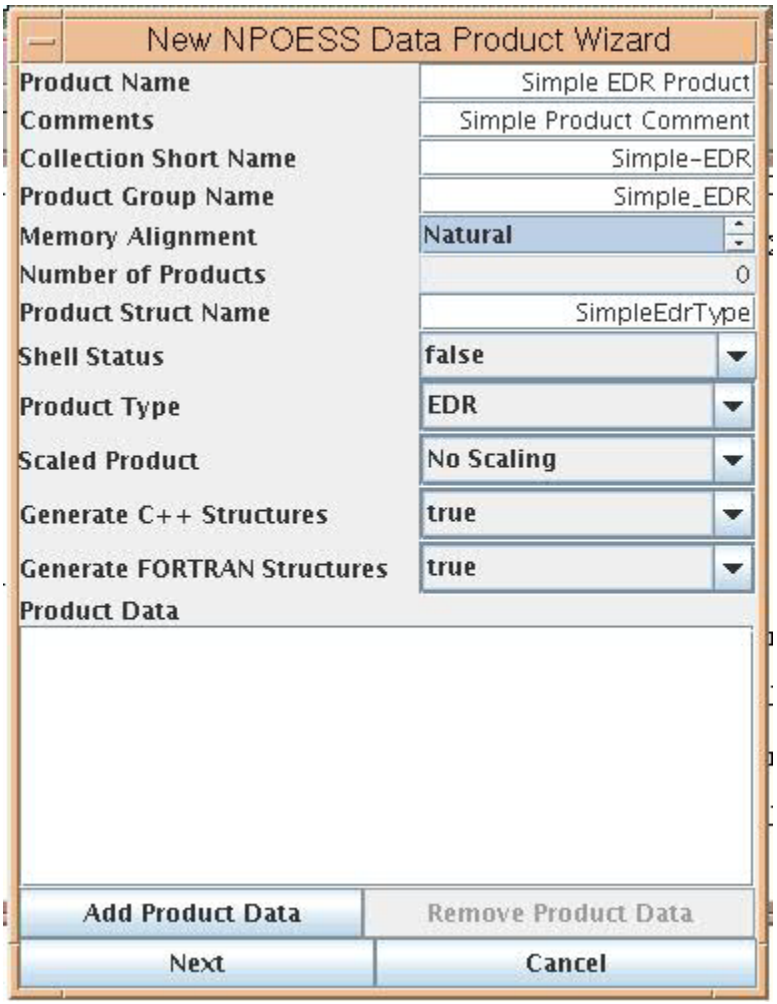

FIGURE 5.2-2 POPULATED DATA PRODUCT WIZARD DIALOG

5.3 PRODUCT DATA POPULATION

In this example there are two fields, so there are two product data entries. To add the first entry the user will click the “Add Product Data” button and the Product Data Wizard as described in section 4.4.2 will open.

The first field is named “Simple Data Array” and the second “Simple Quality Flag”. The first field product data is a Regular type and the second is Quality type. Figure 5.3-1 and Figure 5.3-2 show the populated fields.

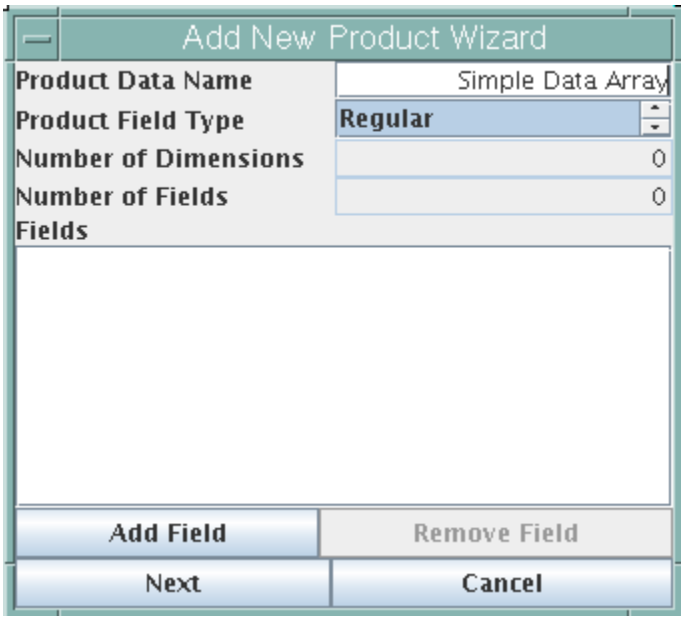

FIGURE 5.3-1 SIMPLE DATA ARRAY PRODUCT DATA

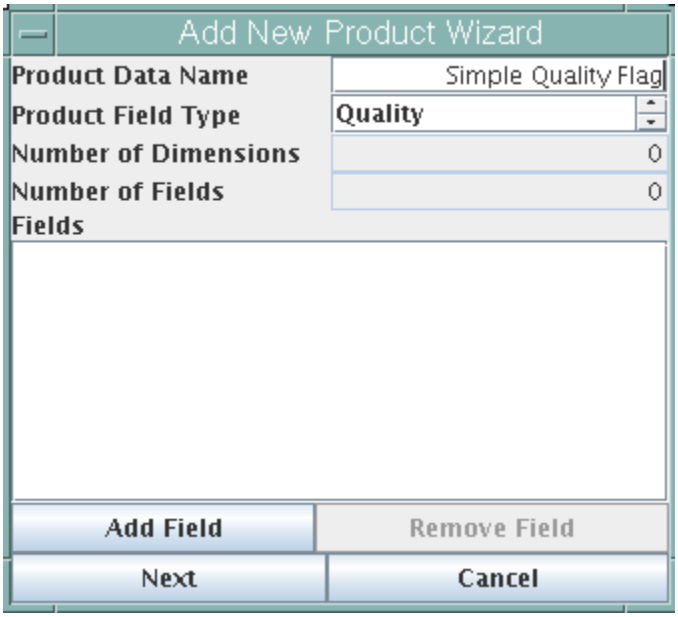

FIGURE 5.3-2 SIMPLE QUALITY FLAG PRODUCT DATA

5.4 FIELD POPULATION

The fields will both be given the same name as the product data. Initial fill for both is zero. And we will not define any dictionary masks. We will fill in the Symbolic Names so that it may be easily refined, if needed, and remain independent of the logical field name. Figure 5.4-1 and Figure 5.4-2 show the populated wizard displays.

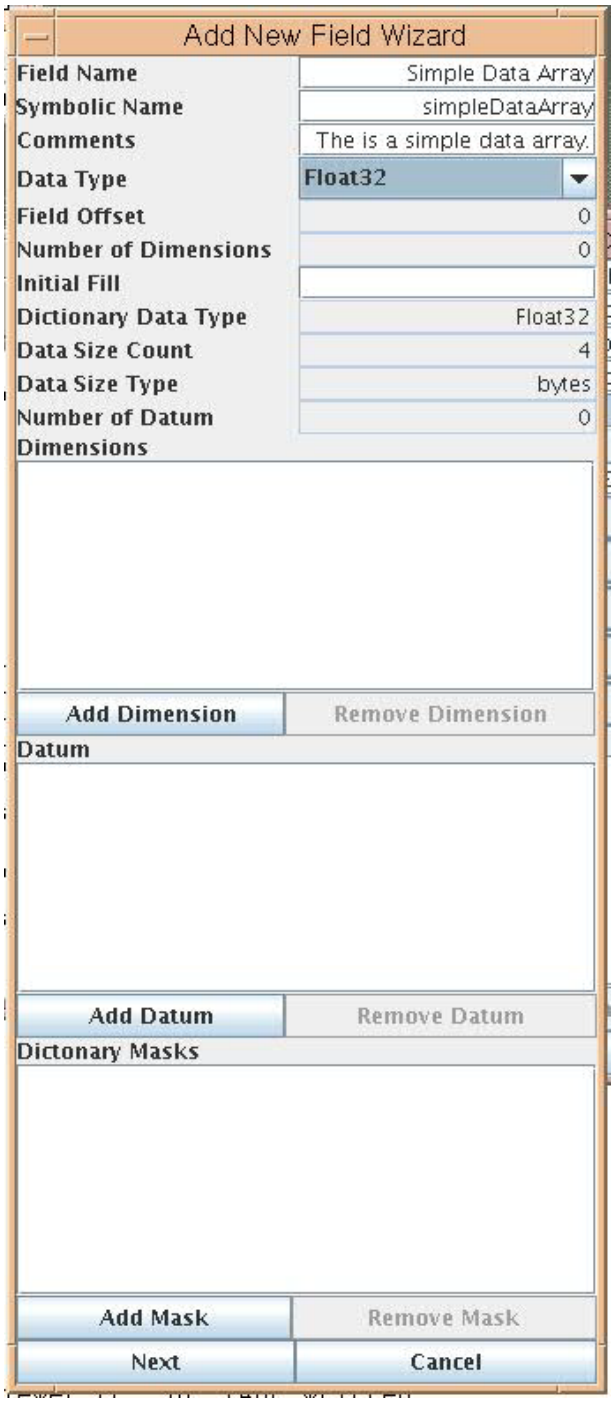

FIGURE 5.4-1 SIMPLE DATA ARRAY FIELD

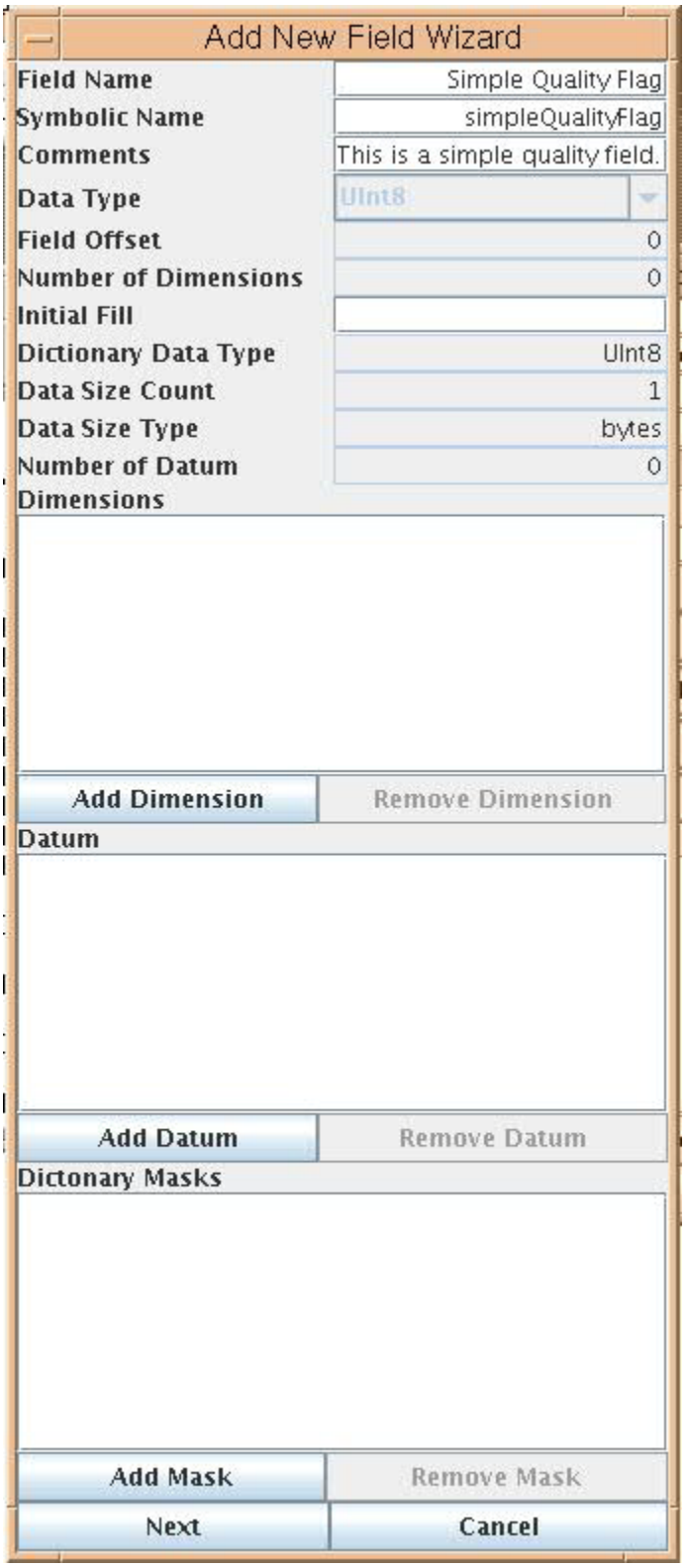

FIGURE 5.4-2 SIMPLE QUALITY FLAG FIELD

5.5 DIMENSIONS POPULATION

Both fields have a simple 10 element dimension. Refer to Figure 5.5-1 for population of the Dimension Wizard. Note that when the Simple Quality Field is defined, the Dimension defined while creating the Simple Data Field may be reused by selecting the appropriate radio button as shown in Figure 5.5-2.

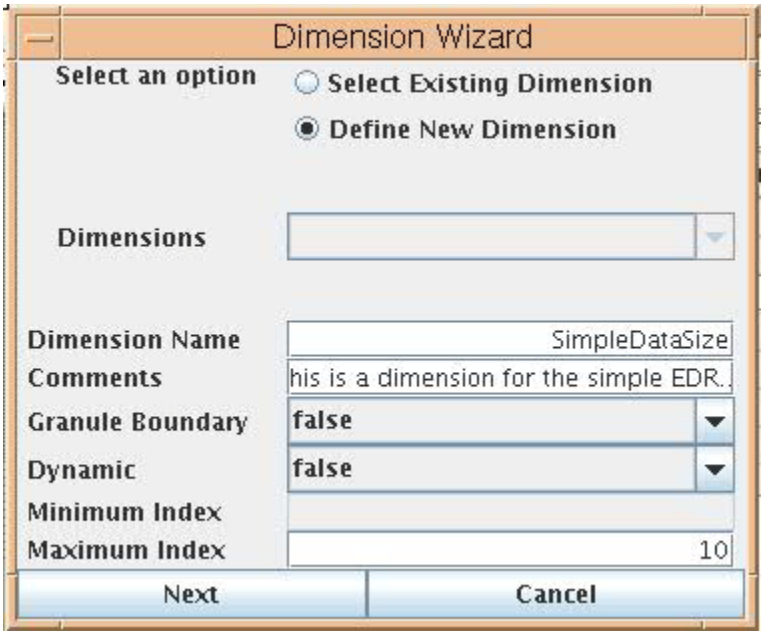

FIGURE 5.5-1 NEW DIMENSION WIZARD

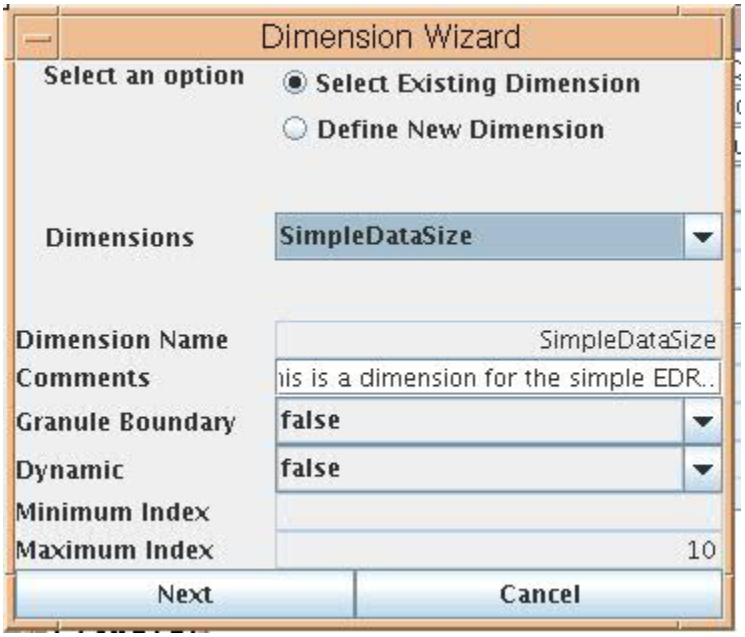

FIGURE 5.5-2 SELECT EXISTING DIMENSION WIZARD

5.6 DATUM POPULATION

For the simple data array, all of the fields are automatically populated except the measurement units, the minimum range, and the maximum range. We will keep the defaults for the fill values and add no new legend entries. We will make the units in meters with a range of 0.0 to 1000.0 as shown in Figure 5.6-1.

The quality flag field will have more than one datum, because there is one for each flag for the quality flag field. We'll create one 1-bit flag and one 2-bit. The 1-bit flag will have one legend entry and the 2-bit flag will have four. Notice that when the 2-bit flag is created, only up to 7 bits are selectable, now, instead of eight. For obvious reasons, the tool will not allow more than eight quality bits in a UInt8 field, except if you are editing an existing quality datum, then it will display a warning message if the eight bits are exceeded after the edit.

For the first datum, enter "Quality Flag 1" for the datum name and select "1 bit(s)" from the "Data Type" combo box. Click on "Add Name" under the Legend Entry Names list area and enter "bit 1" for the name and 0 for the value. Repeat the last step with a name of "bit 2" and value of 1. Click next to close the Legend Entry Wizard and save the values and click Next to close the Datum Wizard and apply the new datum. Refer to Figure 5.6-2 to see the completed Datum. Click Add Datum from the Field Wizard to create the next datum. Enter "Quality Flag 2" for the Datum Name, 0 for the Range Min. Now add four new legend entries with the following values: "bit 1" and 0, "bit 2" and 1, "bit 3" and 2, and "bit 4" and 3. Refer to Figure 5.6-3 to see the finished Datum. Click next to add this datum to the field wizard.

Now click Next to finish the new quality field. Recall that 5 bits are left over. You should see an informative dialog stating that 5 spare quality bits are created. Click "OK". Click next all the way back through to see the new product displayed in a tree Figure 5.6-5. Note that an implicit_pad0 product data has been automatically created to account for the implicit pad bytes in our structure definition. Since each field must be 32-bit aligned (because it is lead by a Float32 field) and the next field is a 1-byte field type (UInt8) with 10 dimensions, there should be 2 pad bytes at the end of the structure. If you expand the implicit_pad node down to the dimension definition (Product Data→ Field → Dimension), you should see that the min and max indexes are both 2.

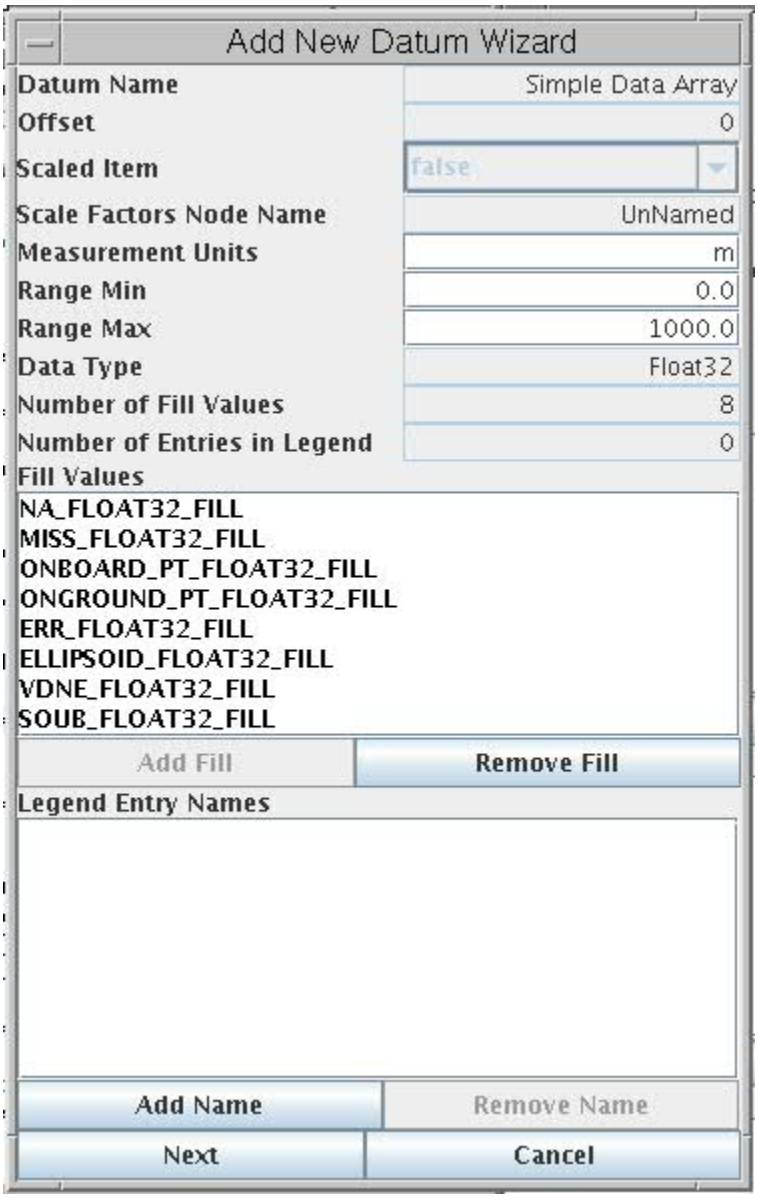

FIGURE 5.6-1 COMPLETED SIMPLE DATA DATUM WIZARD

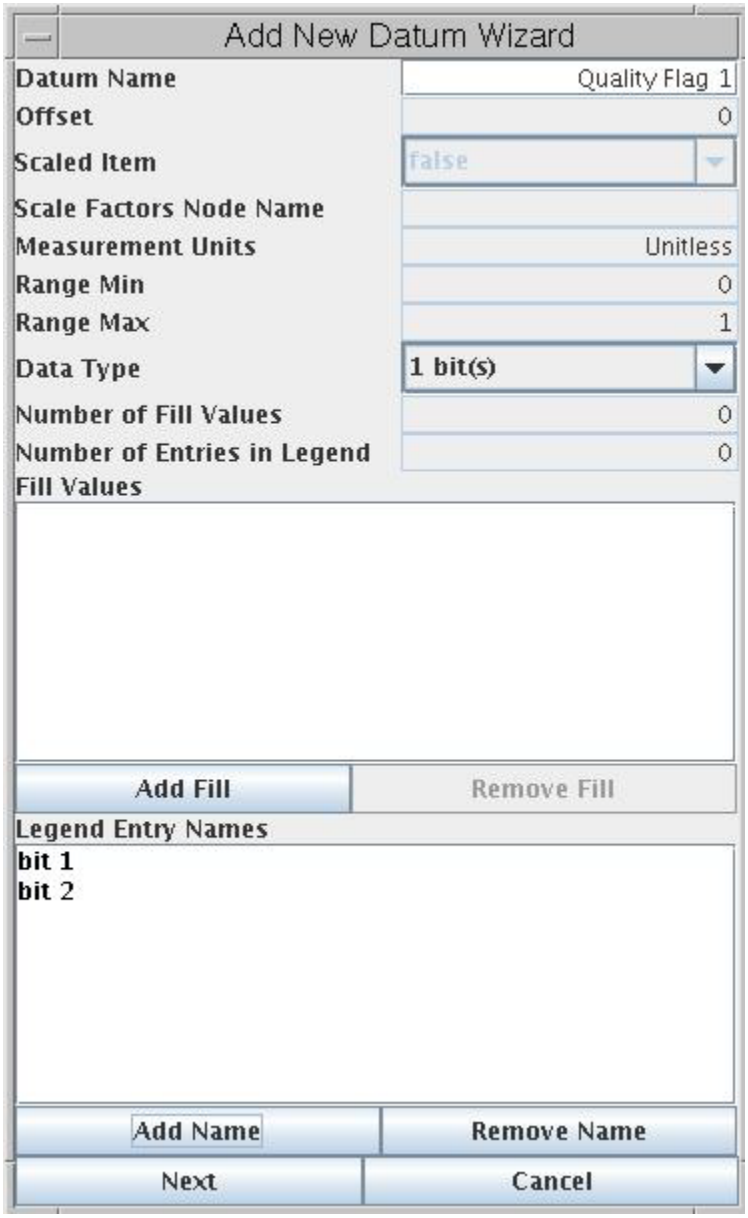

FIGURE 5.6-2 NEW QUALITY FLAG DATUM 1

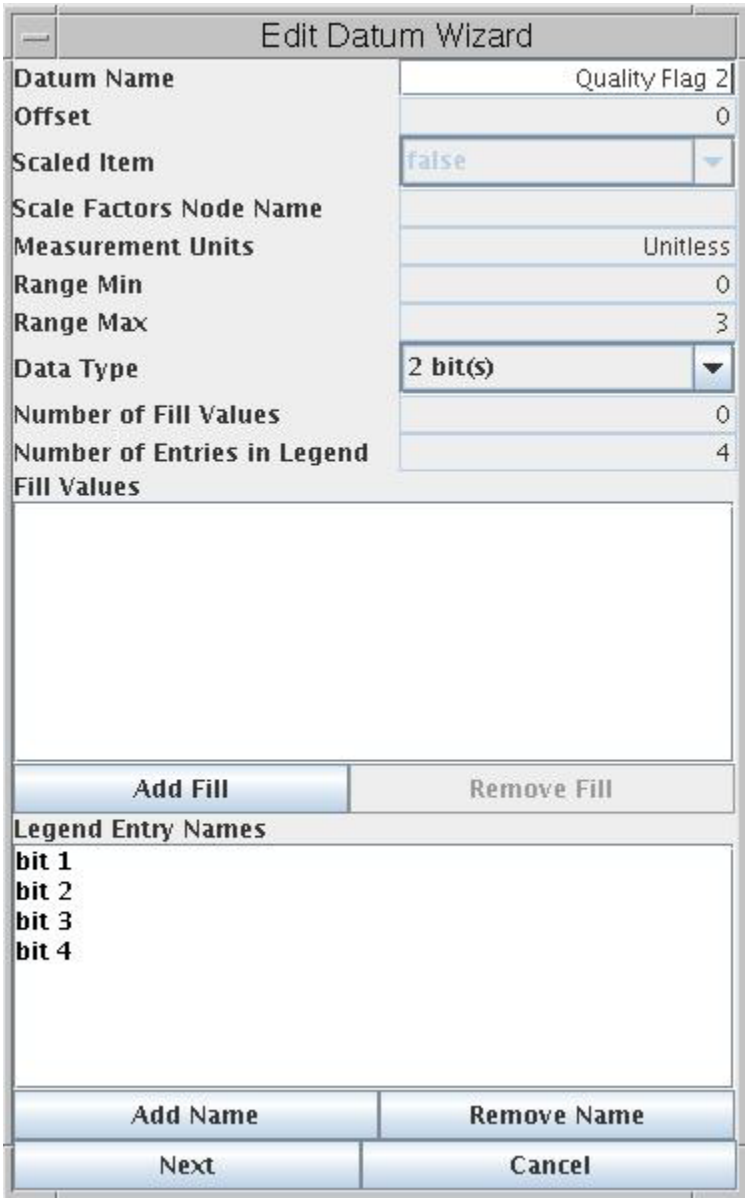

FIGURE 5.6-3 NEW QUALITY FLAG DATUM 2

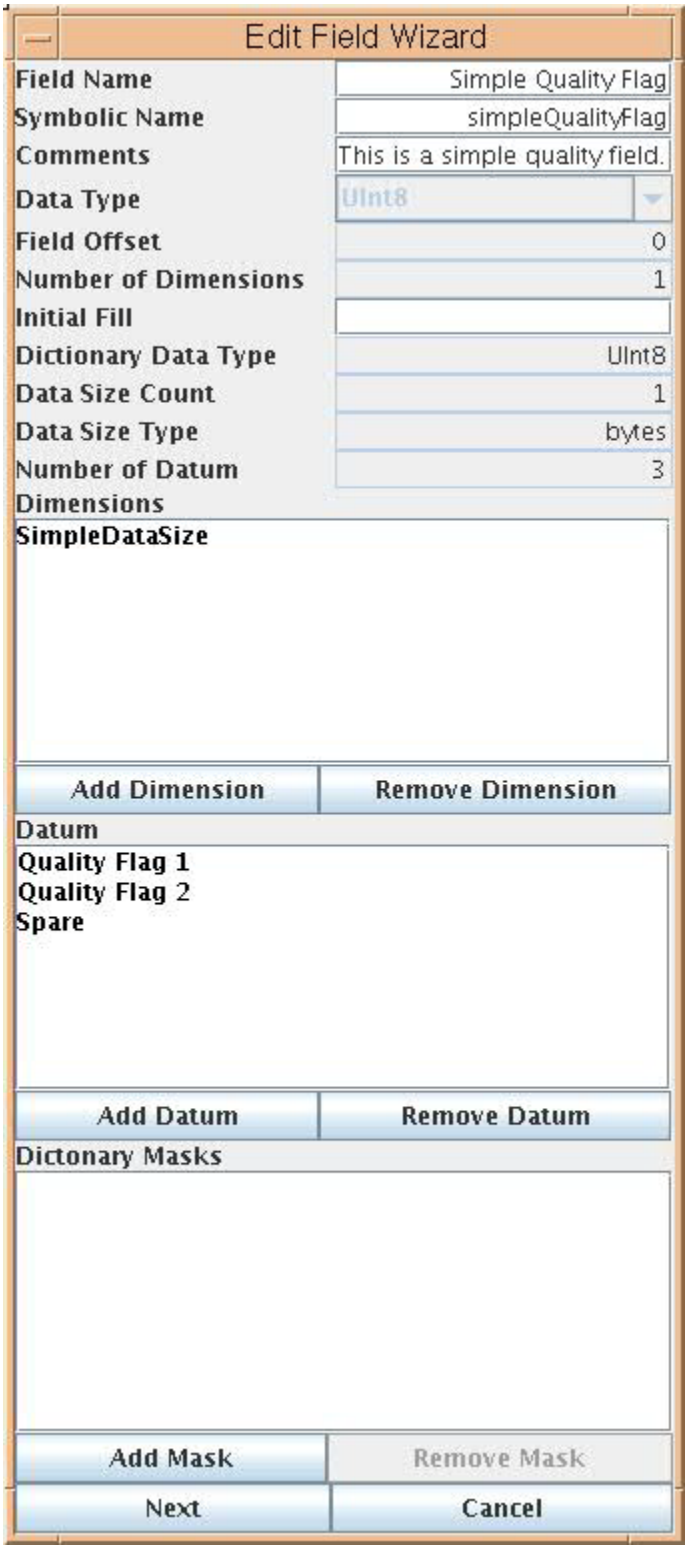

FIGURE 5.6-4 COMPLETED QUALITY FLAG WIZARD

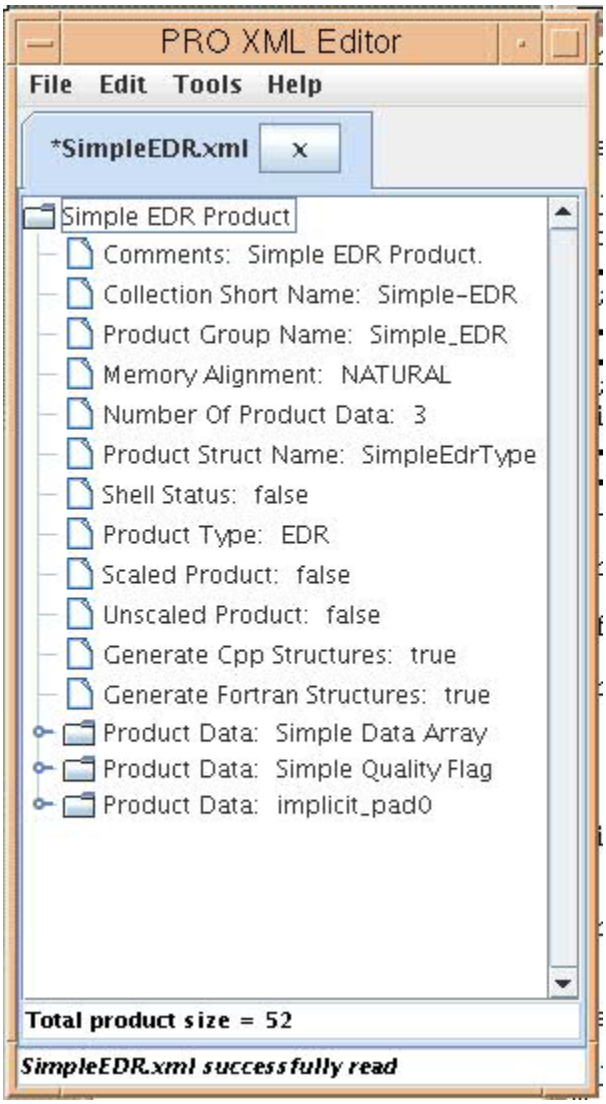

FIGURE 5.6-5 COMPLETED NEW PRODUCT DEFINITION

5.7 SAVING THE NEW PRODUCT

To save the new product, click File and Save. Note that a temporary file has been created at this point, located in the default temp directory of the operating system. On Unix machines, this is usually /tmp. Its name is displayed in the bottom-most text area in the frame. A basic Save dialog is shown. Save the XML as Simple_EDR.xml in your home directory and feel free to examine the contents in its raw form. You will see that the XML has been nicely formatted by the JAXB unmarshaller. You will also see the tab name change to the new xml name. The temporary file that was created earlier should have been deleted immediately after the Save operation. If for some reason the program crashes, the host machine goes down, or some other inevitable failure occurs prior to saving a new product, it may be recovered in the temporary directory. Progress is not saved, however, if the wizard entries have not yet been completed.

6 NOTES

6.1 ACRONYMS AND ABBREVIATIONS

Refer to document LI60917-GND-005 (JPSS CGS Acronyms & Glossary).