IMPORTANT NOTICE: The current official version of this document is available via the Sandia National Laboratories NWMP On-line Documents web site. A printed copy of this document may not be the version currently in effect.

NUCLEAR WASTE MANAGEMENT PROGRAM PROCEDURE

SP 9-5
PARAMETER DATA ENTRY
Revision 0

Effective Date: 02/14/02

Author: Stephen Tisinger *Original signed by Stephen Tisinger* *2/13/02*
(printed name) (signature) date

1.0 Purpose and Scope

This procedure establishes the process for entering into or editing parameter information in the Performance Assessment (PA) Parameter Database (PAPDB). The database is used for both regulatory compliance-level and programmatic PA calculations performed by Sandia National Laboratories' (SNL) Waste Isolation Pilot Plant (WIPP) Project. The procedure ensures parameter information is entered correctly into PA Parameter database and applies to all SNL and contractor personnel who participate in the management of the WIPP PAPDB.

Acronyms and definitions for terms used in this procedure may be found in the Nuclear Waste Management Program (NWMP) Glossary located at the SNL NWMP On-line documents web site.

2.0 Implementation Actions

2.1 Performing the Implementation Actions

The implementation actions described in this section will be effectuated with SQL Server's Query Analyzer tool, a database client that allows SQL statements to be executed against the PAPDB.

Following NP 9-2, *Parameters*, a completed parameter data form, NP Form 9-2-1, is given to the Database Administrator (DBA) to perform the data entry. The DBA should verify that the form is complete in accordance with NP 9-2. In the implementation actions of this section, Form NP 9-2-1 is referred to as "the data entry form" and "the form."

2.2 Creating a Parameter

The DBA shall add a new parameter to the database by completing the following steps:

Step	Description
1	Verify that the parameter does not already exist in the database by attempting to retrieve the parameter specified on the data entry form from the database by executing the following TSQL statement:

Step	Description
	`` select [parameterID] from [parameter] a inner join [material] b on a.[materialID] = b.[materialID] inner join [property] c on a.[propertyID] = c.[propertyID] where b.[abbreviation] = '«material abbreviation»' and c.[abbreviation] = '«property abbreviation »' ``` where «material abbreviation » and «property abbreviation » are the abbreviated names of the material and the property specified on the form. If a match is found, then either (1) an incorrect box was checked on Form NP 9-2-1 or (2) the person initiating the form has unknowingly specified an existing material – property combination and must use a different combination of material and property. In either case, the form shall be returned to the person initiated the form for correction, or for the production of a different form.
2	Attempt to retrieve the ID for the material with the following TSQL statement: ``` select [materialID] from [material] where [abbreviation] = '«material abbreviation»' ``` where «material abbreviation» is the abbreviated name of the material specified on the form. If a match is found, the DBA should make a note of material ID returned and continue on with step 3. If a match is not found, then the DBA – after verifying the material abbreviation was entered correctly – should contact the person initiating the data entry form to get the name and a description for the material abbreviation specified on the form, if not supplied. This new information should be documented on the form, and entered into the database following step (2.1) below.
2.1	For new materials, enter the relevant information for the material with the following TSQL statement:

Step	Description
	<pre>insert into [material] ([isApproved] , [abbreviation] , [name] , [description]) values (1 , '«material abbreviation»' , '«material name»' , '«material description»')</pre> where the items in «» represent the appropriate information given on the form. The value for the [isApproved] column will always be set to integer [1] for all new materials created with this procedure.
2.2	Once the new material has been created, retrieve its ID following step (2) above.
3	Attempt to retrieve the ID for the property with the following TSQL statement: <pre>select [propertyID] from [property] where [abbreviation] = '«property abbreviation»'</pre> where «property abbreviation» is the abbreviated name of the property specified on the form. If a match is found, the DBA should make a note of property ID returned and continue on with step 4. If a match is not found, then the DBA – after verifying the property abbreviation was entered correctly – should contact the person initiating the data entry form to get the name and a description for the property abbreviation specified on the form, if not supplied. This new information should be documented on the form, and entered into the database following step (3.1) below.
3.1	For new properties, complete the following sub-steps (3.1.1 and 3.1.2).
3.1.1	Retrieve the ID for the unit given on the form, if one is specified. Be sure to enter the expression without any spaces.

Step	Description
	`` select [unitID] from [unit] where [abbreviation] = '«unit abbreviation»' ``` where «unit abbreviation» is the unit listed on the form. If an ID is returned, make a note of it before moving on to step 3.1.2. If no ID is returned, run a general query to make sure the unit of measurement is not in the database with different punctuation. ``` select * from [unit] where [abbreviation] like '%«part of unit»%' ``` If no match is found, then enter the new unit with the following TSQL statement: ``` insert into [unit] ([isApproved] , [abbreviation]) values (1 , '«unit abbreviation»') ``` where «unit abbreviation» is the unit listed on the form. Once done, re-execute the following statement to retrieve the unit ID: ``` select [unitID] from [unit] where [abbreviation] = '«unit abbreviation»' ``` where «unit abbreviation» is the unit listed on the form. Make a note of it before moving on to step 3.1.2.

Step	Description
3.1.2	Enter the relevant information for the property with the following TSQL statement: `insert into` `  [property]` `(` `  [unitID]` `, [isApproved]` `, [abbreviation]` `, [name]` `, [description]` `)` `values` `(` `  '«unitID from 3.1.1»'` `, 1` `, '«material abbreviation»'` `, '«material name»'` `, '«material description»'` `)` where the items in «» represent the appropriate information on the form. The value for the [isApproved] column will always be set to integer [1] for all new properties created with this procedure.
3.2	Once the new property has been created, retrieve its ID following step (3) above.
4	Retrieve the appropriate ID for the usage from the parameter type and usage listed on the data entry form. `select` `  a.[usageID]` `from` `  [usage] a` `inner join` `  [parameterType] b` `on` `  a.[parameterTypeID] = b.[parameterTypeID]` `where` `  a.[name] = '« usage»'` `and` `  b.[name] = '«parameter type»'` where the items in « » represent the appropriate information on the form. If no usage ID is returned, break up the above statement into simpler statements and/or verify the spelling of the information entered. If one or more pieces of the information do not exist in the database, refer the form originator to NP 9-2 for a list of applicable usages and/or parameter types. If a new usage is to be entered into the database for this and

Step	Description
	future parameters to be used, complete step (4.1) below; otherwise, continue on to step 5.
4.1	To enter a new usage into the database, a name and a description are needed, along with the type of parameter the new usage should be associated with. After getting a description from the party creating the new usage, enter it into the database with the following statements: `select` `  a.[parameterTypeID]` `from` `  [parameterType] a` `where` `  a.[name] = «parameter type name»` If the usage is applicable to more than one parameter type, execute the above select statement as many times as necessary to get the IDs for each parameter type. Once done, execute the following statement for as many parameter types as are applicable for the new usage. `insert into` `  [usage]` `(` `  [parameterTypeID]` `, [namme]` `, [description]` `)` `values` `(` `, «parameter type ID from step 4»` `, «usage name from the form»` `, «usage description from party creating new usage»` `)`
4.2	Get the ID of the new usage created by completing step (4) above.
5	Create the parameter by entering its information using the following statement: `insert into` `  [parameter]` `(` `  [usageID]` `, [materialID]` `, [propertyID]` `, [isActive]` `, [name]` `, [description]` `)` `values` `(`

Step	Description
	, «usage ID from step 4» , «material ID from step 2» , «property ID from step 3» , 1 , '«parameter name»' , '«parameter description»') where «parameter name» and «parameter description» are from the data entry form. The integer value 1 will always be used for [isActive] for parameters created by this procedure.
6	Follow the process described in the next section, 2.3, Entering or Editing Parameter Values.

2.3 Entering or Editing Parameter Values

2.3.1 Entering Parameter Values

The DBA shall add a new set of parameter values to the database by completing the following steps:

Step	Description
1	Retrieve the ID of the parameter specified on the data entry form from the database by executing the following TSQL statement: select [parameterID] from [parameter] a inner join [material] b on a.[materialID] = b.[materialID] inner join [property] c on a.[propertyID] = c.[propertyID] where b.[abbreviation] = '«material name»' and c.[abbreviation] = '«property name»' where «material name» and «property name» are the abbreviated names of the material and the property specified on the form.
2	Get the ID for the analysis to which the set of values should be assigned: select [analysisID]

Step	Description
	`, [isForCompliance]` `from` `[analysis]` `where` `[abbreviation] = '«analysis abbreviation»'` where «analysis abbreviation» is given on the form. Make a note of the analysis ID and whether or not the analysis is for compliance (a compliance decision) – [Ø] represents *false*, while [1] is equivalent to *true*. If the analysis has not been defined in the database, then the DBA shall contact the PA Department Manager for the appropriate information.
3	Verify that a parameter record with the same set of values does not already exist by executing the following TSQL: `select` `*` `from` `[parameterRecord]` `where` `[parameterID] = «parameter ID from step 1»` If a parameter record with the same set of values already exists for the parameter, relate this to the party who initiated the form. If the party did not realize the values were already in the database, or if the party wishes to change the qualification status of the values from unqualified to qualified, then the form shall be rejected; the information on the form will need to be modified, or a new form will need to be completed in accordance with NP 9-2.
4	Verify that this is the first parameter record for this parameter assigned to the analysis specified on the data entry form. `select` `*` `from` `[parameterRecord]` `where` `[parameterID] = «parameter ID from step 1»` `and` `[analysisID] = «analysis ID from step 2»` If no rows are returned, continue on with step (5). Otherwise, make a note of the ID for the latest parameter record returned (latest being defined as the parameter record assigned to the analysis with the most recent [effectiveDateTime] value). See if the parameter record has been accessed for any analysis (calculation):

Step	Description
	select * from [analysisRetrievalHistory] where [parameterRecordID] = «parameter record ID above» If no rows are returned, then the existing parameter assigned to the analysis can be updated. Jump to the process detailed in section 2.3.2, Parameter Value String Creation and 2.3.4, Parameter Value Replacement.
5	Get the distribution ID for the distribution type for the set of values select [distributionID] from [distribution] where [name] = '«distribution»' where «distribution» is the type of distribution for the values given on the form. If the distribution is not found, then the set of values cannot be used, and a new set of values must be generated using a WIPP-supported distribution type.
6	Have the values for this parameter record been developed under an SNL WIPP QA program? If so use the value [1] for the qualification ID in the next step; if not, use the value [0]. There is a check box near the top of the form (labeled ”Was the data supporting this parameter developed under the SNL WIPP QA Program?”) that will be checked if the values being entered were developed under an SNL WIPP QA program. Only values that have been developed under an SNL WIPP QA program can be entered into the database to be used for a compliance-decision analysis. If a set of parameter values has not been developed under such a program, **it cannot be used in an analysis to make compliance decisions**. If the values are to be used in a compliance decision analysis calculation, and have not been developed under and SNL QA program, the form must be rejected.
7	Add the parameter record to the database: insert into [parameterRecord] ([parameterID] , [analysisID] , [distributionID] , [qualificationID]

Step	Description
	, [isApproved] , [isActive] , [effectiveDateTime] , [value]) values («parameter ID from step 1» , «analysis ID from step 2» , «distribution ID from step 5» , «qualification ID from step 6» , 1 , 1 , getDate() , '«value string (detailed in section **2.3.2**)»') The integer value [1] will always be used to enter or edit the values for [isApproved] and [isActive] in this procedure. The SQL Server function `getDate()` returns the current date and time, and so will stamp the date and time onto a set of parameter values being added.

2.3.2 Parameter Value String Creation

The set of values for a particular parameter record is stored as one string in table [ParameterRecord]. To construct this string, use the appropriate template for the distribution type of the set of values being entered into the database.

Step	Description
1	Get the template to be used with the following TSQL statement: select a.[description] from [template] a inner join [distribution] b on a.[templateID] = b.[templateID] where b.[distributionID] = «distribution ID from form»
2	Fill in the template with the set of values from the form. See one of the sub-steps below (2.1 – 2.3) for further direction, based on the distribution of the value(s).
2.1	Constant distribution The template looks like

Step	Description
	<document><value></value></document> If the constant value on the form is 3.1415926, the value string would be: *'<document><value>**3.1415926**</value></document>'* The above string would be entered into the [value] column for a parameter record.
2.2	Uniform distribution The template looks like *<document> <median></median> <mean></mean> <minimum></minimum> <maximum></maximum> </document>* All of the fields in the template should be filled out with values on the form. Create a string based on the template and the values of the form, and enter it into the [value] column for the parameter record. Note that, while the above template is displayed with spaces and new lines, the string should not contain either of these.
2.3	Other distributions For other distributions, view the appropriate template and fill it in with the data from the form. If there is a question on how to fill out the template, query the database ([ParameterRecord] table) for an existing parameter record of the same distribution type to use as an example. `select` `  a.[value]` `from` `  [parameterRecord] a` `inner join` `  [distribution] b` `on` `  a.[distributionID] = b.[distributionID]` `where` `  b.[distributionID] = «distribution ID from form»`

2.3.3 Parameter Value Replacement

A set of values for a parameter is assigned to a particular analysis. If this set of values changes before it is accessed (retrieved by SDBREAD_LIB), there is no need to keep a record of the latest

set of values for that analysis. In this case, rather than inserting a new parameter record (row into table [ParameterRecord]), one or more attributes for an existing record can be updated.

Step	Description
1	Get the distribution ID for the distribution type for the set of values `select` `  [distributionID]` `from` `  [distribution]` `where` `  [name] = '«distribution from form»'` If the distribution is not found, then the set of values cannot be used, and a new set of values must be generated using a WIPP-supported distribution type.
2	Have the values for this parameter record been developed under an SNL WIPP QA program? If so use the value [1] for the qualification ID in the next step; if not, use the value [0]. There is a check box near the top of the form (labeled ”Was the data supporting this parameter developed under the SNL WIPP QA Program?”) that will be checked if the values being entered were developed under an SNL WIPP QA program. Only values that have been developed under an SNL WIPP QA program can be entered into the database to be used for a compliance-decision analysis. If a set of parameter values has not been developed under such a program, **it cannot be used in an analysis to make compliance decisions**. If the values are to be used in a compliance decision analysis calculation, and have not been developed under and SNL QA program, the form must be rejected.
3	Construct a parameter value string as detailed in section 2.3.2, Parameter Value String Creation, above.
4	Update the parameter record with the following TSQL: `update` `  [parameterRecord]` `set` `  [distributionID] = «ID from step 1»` `,  [qualificationID] = «ID from step 2»` `,  [isApproved] = 1` `,  [isActive] = 1` `,  [effectiveDateTime] = getDate()` `,  [value] = «string from step 3»` `where` `  [parameterRecordID] =` `  «parameter record ID from step 6 of section **2.3.1**»`

2.4 Adding References (Justification Documents)

Often, the same reference is used for multiple parameters or parameter records; therefore, it is often that a new reference for a particular parameter will already exist in the database – in this case, only a link needs to be created between the parameter and the reference, and a pointer to the applicable part(s) of the document for the parameter should be noted. Repeat the steps below for each reference / document to be added / associated with the parameter.

Step	Description
1	Attempt to retrieve the reference / document. Search for part of the document's title, or on the last name of an author, for example. `select` `  *` `from` `  [Document]` `where` `  [citation] like '%«keyword in title»%'` or `select` `  *` `from` `  [Document]` `where` `  [citation] like '%«last name of an author»%'` If one or more rows are returned, compare the citation to that given on the data entry form. If there is a match with one of the rows returned, make a note of the ID of the document.
2	If there was no match from searching for both a keyword and on a last name, add a new document to the table. `insert into` `  [Document]` `(` `  [documentTypeID]` `, [documentRoleID]` `, [locationTypeID]` `, [location]` `, [citation]` `)` `values` `(` `  1` `, 0` `, 1` `, ''` `, '«citation»'` `)` where «citation» is a string that matches the particular citation string

Step	Description
	template for the type of document being entered. Use an existing citation in the database as a guide if needed. After adding the new document, get its ID by performing one of the queries described in step (1) above.
3	Verify that the document is not already associated with the parameter, or one of its particular records. If a new document / reference was added in step (2) above, there is no need to perform this verification. `select` `  *` `from` `  [Documentation]` `where` `  [documentID] = «document ID»` where «document ID» is the ID for the document found in step (1) or step (2) above. This list may be quite extensive. If a shorter list is desired, one can issue a query with more specific criteria: `select` `  *` `from` `  [documentation]` `where` `  [documentID] = «document ID»` `and` `  (` `  [entityKeyID] = «parameter ID»` `  or` `  [entityKeyID] = «parameter record ID»` `  )` where «parameter ID» and «parameter record ID» are the IDs of the parameter and the parameter record, respectively. These IDs are found by executing one or more of the queries described in sections **2.2** and **2.3**. If the document is already associated with the parameter or parameter record, verify that the column specifying the location in the document ([locationInDocument]) is accurate. If not, update this column with a TSQL statement similar to the following: `update` `  [documentation]` `set` `  [locationInDocument] = '«page / location info»'` `where` `  documentationID = «documentation ID»` where «page / location info» is a description of where the parameter or parameter record is referenced in the document / reference, and

Step	Description
	«documentation ID» is the ID of a particular row returned earlier in this step.
4	If the document was not associated with the parameter or parameter record, add a new row to table [Documentation]. `insert into` `  [documentataion]` `(` `  [documentID]` `, [entityTypeID]` `, [entityKeyID]` `, [locationInDocument]` `)` `values` `(` `  «document ID»` `, «entity type ID»` `, «entity key ID»` `, «page / location info»` `)` In order to complete this insert statement, the entity type must be known. To find the appropriate «entity type ID», issue the following TSQL statement: `select` `  *` `from` `  [entity]` The most commonly used entity types are “parameter” (ID of [1]) and “parameter record” (ID of [2]). For the «entity key ID» attribute, enter the ID that corresponds to the type of association being made. If the document is being associated with the parameter, enter the ID of the parameter for the «entity key ID»; if the document is relevant only to a particular parameter record, enter the ID of that parameter record for the «entity key ID».

2.5 Adding or Modifying Analyses

Before parameter records can be added to the database, the analysis with which the records should be associated must exist.

Step	Description
1	Verify that the analysis does not exist `select`

Step	Description
	* from [analysis] Visually verify that the analysis does not exist. If it does exist, make a note of its ID ([analysisID]).
2	If the analysis does not exist, create it: insert into [analysis] ([valueTypeID] , [isForCompliance] , [abbreviation] , [name] , [description]) values («value type ID» , {0 or 1} , '«abbreviation»' , '«name»' , '«description»') where «abbreviation», «name» and «description» are given by the PA Department Manager. If the analysis is for a compliance decision, then [1] should be entered for [isForCompliance]; otherwise, [0] should be entered. If no default type of value is assigned to the analysis, [0] should be entered for the «value type ID». Otherwise, lookup the ID for the type of value with this TSQL: select * from [valueType] where [name] = '«type of value»' where «type of value» is given by the PA Department Manager as applicable (e.g., median, mean).
3	If the analysis exists, verify its information is correct. If not, execute an update statement, as appropriate.

2.6 Adding or Modifying Computational Codes Associated with an Analysis

In order for computational codes to retrieve parameter values, specific versions of the codes must be assigned to an analysis.

Step	Description
1	Get the ID of the computational code: `select` `  *` `from` `  [computationalCode]` Make a note of the ID that matches a particular code. If the code is not found, add it to the database: `insert into` `  [computationalCode]` `(` `  [name]` `,  [description]` `)` `values` `(` `  '«name of the code»'` `,  '«description of the code»'` `)` where the code's name and description are provided by the PA Department Manager..
2	Get the ID of the analysis to which the code is to be assigned: `select` `  *` `from` `  [analysis]` Make a note of the ID of the appropriate analysis. If the analysis is not returned, follow the process described in section 2.5, Adding or Modifying Analyses.
3	Verify that the code is not already associated with the analysis. `select` `  *` `from` `  [analysisComputationalCode]` `where` `  [analysisID] = «analysis ID»` `and` `  [computationalCodeID] = «computational code ID»` where the IDs are from steps (1) and (2) above. If no rows are returned, proceed to step (4). Otherwise, confirm that the version of the

Step	Description
	code matches that given on the form. If the versions are not identical, update the value in the table: `update` `  [analysisComputationalCode]` `set` `  [computationalCodeVersion] = «version»` `where` `  [analysisID] = «analysis ID»` `and` `  [computationalCodeID] = «computational code ID»` where «version» is given by the PA Department Manager. The IDs are from steps (1) and (2) above.
4	If the code is not associated with the analysis (see step (3)), add the association to the database: `insert into` `  [analysisComputationalCode]` `(` `  [analysisID]` `,  [computationalCodeID]` `,  [computationalCodeVersion]` `)` `values` `(` `  «analysis ID»` `,  «computational code ID»` `,  «computational code version»` `)` where the version is from the PA Department Manager, and the IDs are from steps (1) and (2) above.

2.7 Safety

Activities covered by this procedure involve only those hazards associated with normal office activities and are assessed in PHS SNL9A00194, Center 6800 Office Space. The hazards are adequately mitigated by the requirements that all personnel be current in all SNL required basic ES&H training.

3.0 Records

No records are generated for or by this procedure.

4.0 Appendices

Not applicable. No forms or flow charts are used in conjunction with this procedure.