

# XML Documentation

---

## Introduction

This documentation is intended for an experienced programmer to get Design-Expert<sup>®</sup> version 7 software (DX7) up and running in conjunction with automated experiment systems. Extensible Markup Language (XML) is DX7's batch mode input and output. If you do not consider yourself an "XML capable" programmer please visit <http://www.xmlfiles.com/xml/>. This site is useful even if you are an *XMLpert*.

## Definitions

Throughout this document, references to XML scripts, XML documents, and programs will crop up. To prevent confusion these items will be defined as follows:

An "XML script" is a set of instructions for another program to execute. Design-Expert accepts a set of commands via XML scripts. These commands are detailed in the **DX7 XML Scripting Commands** section found later on in this document. The only script DX7 generates is an XML design script when you Save a design as XML. DX7 can use this type of script as both input and output. This file is really an XML document but can be processed by DX7 as a script.

An "XML document" is just a tag delimited text file. DX7 generates XML documents of the various reports when you use the various "Export as" menu commands. The exported reports include the Design Summary, the Design Evaluation, ANOVA, and diagnostics, as well as the output of the numerical optimization and point prediction nodes. In short, if DX7 displays text, it can be exported to an XML document. An Exported XML design document cannot be used by DX7 as input.

A "program" gets instructions from handles, user input, scripts and/or output from other programs. Generally, programs process input into some kind of output. The programs referenced in this document are Design-Expert, your automated system, and a translator to allow Design-Expert and your system to talk to one another.

"Document type declarations" (DTD's) may creep into the conversation. DTD's are just maps for XML parsing programs to display and validate XML documents. You do not need DTD's in order to use XML, but they are nice to have in order to view XML.

*Undetectable errors are infinite in variety, in contrast to detectable errors, which by definition are limited.*

*- [Laws of Computer Programming, IX](#)*

*The best way to prepare [to be a programmer] is to write programs, and to study great programs that other people have written. In my case, I went to the garbage cans at the Computer Science Center and fished out listings of their operating system.*

*- [Bill Gates](#)*

## XML Scripting – Getting Started

Scripts can be executed by Design-Expert in a number of ways. The easiest is to use the File → Open menu option. You can also drag and drop the script file on to a Design-Expert shortcut. Finally, Design-Expert can be started via the Start → Run selection and the script executed as a handle.

```
"C:\Program Files\DX7\Dx7.exe" "C:\ Program Files\DX7\starting design.xml"
```

The first part starts Design-Expert; the second part indicates the script to execute. The quotes are required unless all PATHS have been defined in your operating system.

Scripts are executed as soon as they are read in if the status attribute is set to “ON.” Scripts that are read in but not executed can be run using the Run Script command (Ctrl-F5). They can also be halted (Shift-F5) or killed (Ctrl-Shift-F5). When running a script, warning and error messages will not appear on the screen but will be output to a log file if one is specified in the script tag.

A script is a series of one or more script lines. A script line is a command with an optional integer and/or string argument. The integer typically identifies the response the command is supposed to operate on. The string argument is often an input or output file name. The script line may include additional script data as well. The response index and file name arguments can alternatively be passed as scriptData elements. For example, the following command exports the ANOVA report for the second response in the design to a file in the same directory called “YieldAnova.xml”:

```
<!DOCTYPE dex7XmlDoc SYSTEM "dex7XmlDoc.DTD">  
<dex7XmlDoc name="exportanova.xml" creator="DX7">  
<scriptLine command="ExportAnova" integerArg="2" stringArg="YieldAnova.xml" />  
</dex7XmlDoc>
```

This is the same command using the scriptData elements to pass the arguments:

```
<!DOCTYPE dex7XmlDoc SYSTEM "dex7XmlDoc.DTD">  
<dex7XmlDoc name="exportanova.xml" creator="DX7">  
<scriptLine command="ExportAnova">  
<scriptData>  
<responseIndex>2</responseIndex>  
<fileNameData>YieldAnova.xml</fileNameData>  
</scriptData>  
</scriptLine>  
</dex7XmlDoc>
```

Script commands are processed sequentially in the order as they appear in the XML file with the starting state for a script command being whatever state the previous command left the program in. Element tag names and attribute names are case sensitive.

The export commands will create a file at the location you specify, if the file doesn't already exist. If the file exists at that location, it will be overwritten. You may alter this behavior by adding "&gt;&gt;" to the beginning of the path (remove the quotes). This will cause the exported information to be appended to the file. The following script will cause ANOVA's for response 2 and response 3 to be written to the Anova.xml file.

```
<!DOCTYPE dex7XmlDoc SYSTEM "dex7XmlDoc.DTD">
<dex7XmlDoc name="exportanova.xml" creator="DX7">
<scriptLine command="ExportAnova">
<scriptData>
<responseIndex>2</responseIndex>
<fileNameData>Anova.xml</fileNameData>
</scriptData>
</scriptLine>
<scriptLine command="ExportAnova">
<scriptData>
<responseIndex>3</responseIndex>
<fileNameData>&gt;&gt;Anova.xml</fileNameData>
</scriptData>
</scriptLine>
</dex7XmlDoc>
```

It is suggested that you use the more verbose scriptData method to write your scripts as they are much easier to debug and change if you do. The whole point of XML is to let everyone know what is intended by giving them a tag denoting what the information means.

## DX7 XML Scripting Commands

### OpenDesignFile

Opens a design file. Equivalent to selecting the file using File => Open from the graphical interface. The string argument indicates the file that will be read.

### ExportDesign

Exports the currently loaded design to an XML document specified by the string argument. This will export the factor info, block info, design model, and design rows. This command creates an XML design script which can be used as both input and output to DX7.

### EndSession

Closes and exits the program after completing any pending commands.

The following example script opens a Design-Expert 7 file, saves it as an XML script then closes Design-Expert 7. This is useful for converting existing designs to an XML script. It is also useful to avoid the need to remember to Save As XML.

```
<!DOCTYPE dex7XmlDoc SYSTEM "dex7XmlDoc.DTD">
<dex7XmlDoc name="create.xml" creator="DX7">
<script status="ON" logFile="out.log">
<scriptLine command="OpenDesignFile" stringArg="starting design.dx7" />
<scriptLine command="ExportDesign" stringArg="starting design.xml" />
<scriptLine command="EndSession" />
</script>
</dex7XmlDoc>
```

## **ResponseAnalysisSelectionType \***

Sets the type of analysis for the response specified by the integer argument. The analysis type can be set by using the modelSelection scriptdata element. See Command Details section. You may also set the analysis type by setting the string argument to “Forward” “Stepwise” or “Backward”. Using this method the alpha in and alpha out will default to 0.1.

## **AnalyzeResponse**

Analyzes the response specified by the integer argument. The model to be analyzed may be selected either by the SetResponseModel or ResponseAnalysisSelectionType commands or by specifying it in the selectionMethod element of the design responseModel using the baseModel and selectionMethod attributes. If not specified, the model defaults to the order recommended by the Fit Summary for polynomial analyses or to the intercept-only model in the factorial case.

## **SetResponseModel**

Sets the selected model for the response specified by the integer argument. The model should be specified in the string argument by enumerating the effects desired separated by spaces. The terms should be represented by single letters A to Z. For example, to select some effects from a 4 factor RSM you could use the string “A B C AB AC AD A^2”.

## **SetResponseTransform**

Sets the transform for the response specified by the integer argument. The transform should be specified in the string argument using the same name as it appears in the program. Any numeric arguments to the transform (k, lambda, logit bounds) should be put after the transformation name separated by semi-colons. Example: “Logit;0;100”.

## **ExportAnalysis**

Analyzes the response specified by the integer argument and exports a report to an XML document specified by the string argument. For factorial analyses, the report consists of an Effects report followed by an ANOVA. For polynomial analyses, the Fit Summary and ANOVA are reported.

The model to be analyzed may be selected either by the SetResponseModel or ResponseAnalysisSelectionType commands or by specifying it in the selectionMethod element of the design responseModel using the baseModel and selectionMethod attributes. If not specified, the model defaults to the order recommended by the Fit Summary for polynomial analyses or to the intercept-only model in the factorial case.

## **ExportPointPrediction**

Exports the Point Prediction report to an XML document specified by the string argument. Use the codedPoint or actualPoint tags within the scriptData to specify the point to predict. If no point is given, the report will use the centroid.

## **ExportEvaluation**

Exports the Design Evaluation report to an XML document specified by the string argument.

## **ExportBuildEvaluation**

Exports the Build Evaluation report to an XML document specified by the string argument. The Build Evaluation report only appears on certain factorial designs (2-Level, Taguchi). You must build the design to get the report by specifying either an already built design or a script with a buildable design in the inputNameFileData tag.

## **ExportEffects**

Exports the Effects report for the response specified by the integer argument to an XML document specified by the string argument.

## **ExportFitSummary**

Exports the Fit Summary report for the response specified by the integer argument to an XML document specified by the string argument.

## **ExportAnova**

Exports the ANOVA report for the response specified by the integer argument to an XML document specified by the string argument.

## **ExportDiagnostics**

Exports the Diagnostics Case Statistics report for the response specified by the integer argument to an XML document specified by the string argument.

## **ExportOptiNumCriteria**

Exports the Numerical Optimization criteria to an XML document specified by the string argument.

## **ExportOptiNumSolutions**

Exports the Numerical Optimization solutions to an XML document specified by the string argument.

## **ExportOptiGraphCriteria**

Exports the Graphical Optimization criteria to an XML document specified by the string argument.

## **ExportConstraints**

Exports the Constraints report to an XML document specified by the string argument.

**SetPreference \***

Uses stringArg to set the preference. The string is in the format “[Area]prefItemString=newSetting.”

For example, [MathAnalysis]CodedMaximum=10.”

**BuildDesign \***

Builds a design using the design information provided in the same file or in the file specified by stringArg. When building, row information normally provided in the design data is not needed but additional information needed for building that may be needed is provided in the buildInfo tag. (see buildInfo tag attributes)

**StartBuild**

Same as BuildDesign only stops the build process on the first screen and allows the user to complete the build on his or her own.

**ExportSummary**

Exports the Design Summary report to an XML document specified by the string argument.

**ImportDesign**

Same function as OpenDesignFile.

**LoadXmlDoc**

Synonym for ImportDesign. (code is identical).

\* see **Command Details** section.

## Command Details

This section covers the attributes of some of the larger and/or more commonly used DX7 XML commands. For instance, the BuildDesign command allows the user to set every option for every type of design.

### ResponseAnalysisSelectionType

Use a modelSelection scriptData phrase within ResponseAnalysisSelectionType to set the details about what DX7 is to automatically analyze. Use this command before AnalyzeResponse or ExportAnalysis to make sure you are starting with the model you want.

- baseModel: picks the full model from which selection is to begin.
  - Options include: Linear, 2FI, Quadratic, Cubic, Quartic, Fifth, Sixth
- selectionMethod: allows the choice of Backward, Forward, or Stepwise
- alphaIn: sets the include p-value for a coefficient.
- alphaOut: sets the remove p-value for a coefficient.

Additionally for mixtures:

- baseModel:
  - Options include: Linear, Quadratic, Special Cubic, Cubic, Special Quartic
- modelType: choose from Scheffe (default), Cox, or Slack
- calcType: choose from pseudo, real, or actual

The following script opens “dataxmlout.xml” analyzes the first response using a Quadratic model, with Backward elimination, using the default 0.1 elimination p-value. It then goes on to analyze the design and export the full analysis and the diagnostics report.

```
<!DOCTYPE dex7XmlScript SYSTEM "dex7XmlDoc.DTD">
<dex7XmlScript name="RsmAnalyze.xml" creator="Dx7">
<script status="ON" logFile="out.log">
<scriptLine command="OpenDesignFile" stringArg="dataxmlout.xml" />
<scriptLine command="ResponseAnalysisSelectionType" integerArg="1">
<scriptData>
<modelSelection baseModel="Quadratic" selectionMethod="Backward" />
</scriptData>
</scriptLine>
<scriptLine command="ExportAnalysis" integerArg="1"
stringArg="R1AnalysisReport.xml" />
<scriptLine command="ExportDiagnostics" integerArg="1"
stringArg="R1Diagnostics.xml" />
<scriptLine command="ExportDesign" stringArg="analyzed design.xml" />
<scriptLine command="EndSession"/>
</script>
</dex7XmlScript>
```

## BuildDesign

### **Two-Level Factorial** (Factorial2)

- noOfBaseFactors – The number of factors that make up the baseFactors tag
- baseFactors – Factors used in the factor generators (e.g. “ABD”)
- noOfCenterPoints
- noOfFactorialReps
- The build info tag for 2-level factorials also has child tags factorGenerator and blockGenerator.
- factorGenerator has attributes “factor” and “generator”, which designates the factor letter and the generator for that factor
- blockGenerator has attributes “block” and “generator” which specify the block number and the generator for that block

### **Min Run Res V** (MinResV)

- noOfCenterPoints

### **Min Run Res IV** (MinResIV)

- noOfCenterPoints
- extraMinResIVRuns – “true” or “false”

### **Irregular Fraction** (IrregularFraction)

- noOfCenterPoints

### **General Factorial** (Factorial)

- noOfReplicates

### **Factorial D-Optimal** (DOptimal)

- processOrder – “Main Effects”, “2FI”
- coordinateExchange – “true” or “false”
- noOfLackOfFit
- noOfReplicates
- noOfCenterPoints
- forceCategoricBalance - “true” or “false”
- penalty - Positive value  $\leq 1$

### **Plackett-Burman** (PlackettBurman)

- noOfCenterPoints

### **Taguchi OA** (Taguchi)

### **Central Composite** (CCD)

- noOfCenterPoints
- noOfAxialReplicates
- noOfCtrPerAxialBlk
- alphaType – “Rotatable”, “Spherical”, “Orthogonal Blocks”, “Orthogonal Quadratic”, “Practical”, “Face Centered”, “Other” (specify alpha)
- alpha – Value between .10 and 6 (will override the alphaType)
- CCDType – “Full”, “Small”, “1/2 Fraction”, “Min Run Res V”
- \* Center points per factorial block will be the noOfCenterPoints – noOfAxialReplicates

**Box-Behnken** (BoxBehnken)

- noOfCenterPoints

**One Factor** (OneFactor)

- noOfCenterPoints
- processOrder – “Linear”, “Quadratic”, “Cubic”

**Three-Level Factorial** (3LvlFactorial)

- noOfCenterPoints

**Hybrid** (Hybrid)

- noOfCenterPoints

**Pentagonal** (Pentagonal)

- noOfCenterPoints

**Hexagonal** (Hexagonal)

- noOfCenterPoints

**RSM D-Optimal** (DOptimal)

- processOrder – “Main Effects”, “Linear”, “2FI”, “Quadratic”, “Cubic”
- noOfLackOfFit
- noOfReplicates
- noOfCenterPoints
- coordinateExchange – “true” or “false” (“false” will use point exchange with vertices, centers of edges, axial check points, interior points, and the overall centroid as candidates)
- forceCategoricBalance - “true” or “false”
- penalty - Positive value  $\leq 1$

**RSM Distance Based** (DistanceBased)

- processOrder – “Main Effects”, “Linear”, “2FI”, “Quadratic”, “Cubic”
- noOfLackOfFit
- noOfReplicates
- noOfCenterPoints
- coordinateExchange – “true” or “false” (“false” will use point exchange with vertices, centers of edges, axial check points, interior points, and the overall centroid as candidates)
- forceCategoricBalance - “true” or “false”
- penalty - Positive value  $\leq 1$

**RSM User Defined** (UserDefined)

- processOrder – “Main Effects”, “Linear”, “2FI”, “Quadratic”, “Cubic”
- noOfLackOfFit
- noOfReplicates
- noOfCenterPoints

**RSM Historic** (Dummy) {builds an empty design}

- rows

**Simplex Lattice** (SimplexLattice)

- mixOrder – “Linear”, “Quadratic”, “Special Cubic”, “Cubic”
- augmentDesign – “true” or “false”
- noOfReplicates

**Simplex Centroid** (SimplexCentroid)

- mixOrder – “Linear”, “Quadratic”, “Special Cubic”, “Cubic”
- augmentDesign – “true” or “false”
- noOfReplicates

**Screening** (SimplexScreen or NonSimplexScreen)

- noOfCenterPoints
- noOfVerticesDesired
- includeAxialCB – “true” or “false”
- includeCentroids – “true” or “false”

**Mix D-Optimal** (DOptimal)

- mixOrder – “Linear”, “Quadratic”, “Special Cubic”, “Cubic”
- coordinateExchange – “true” or “false” (“false” will use point exchange with vertices, centers of edges, axial check points, interior points, and the overall centroid as candidates)
- noOfLackOfFit
- noOfReplicates
- noOfCenterPoints

**Mix User Defined** (UserDefined)

- mixOrder – “Linear”, “Quadratic”, “Special Cubic”, “Cubic”
- noOfLackOfFit
- noOfReplicates
- noOfCenterPoints

**Mix Distance** (DistanceBased)

- mixOrder – “Linear”, “Quadratic”, “Special Cubic”, “Cubic”
- coordinateExchange – “true” or “false” (“false” will use point exchange with vertices, centers of edges, axial check points, interior points, and the overall centroid as candidates)
- noOfLackOfFit
- noOfReplicates
- noOfCenterPoints

**Mix Historic** (Dummy) {builds an empty design}

- rows

### **Combined D-Optimal** (DOptimal)

- processOrder – “Main Effects”, “Linear”, “2FI”, “Quadratic”, “Cubic”
- mixOrder – “Linear”, “Quadratic”, “Special Cubic”, “Cubic”
- mix2Order – “Linear”, “Quadratic”, “Special Cubic”, “Cubic”
- coordinateExchange – “true” or “false” (“false” will use point exchange with vertices, centers of edges, axial check points, interior points, and the overall centroid as candidates)
- noOfLackOfFit
- noOfReplicates
- noOfCenterPoints
- forceCategoricBalance - “true” or “false”
- penalty - Positive value  $\leq 1$

### **Combined User Defined** (UserDefined)

- processOrder – “Main Effects”, “Linear”, “2FI”, “Quadratic”, “Cubic”
- mixOrder – “Linear”, “Quadratic”, “Special Cubic”, “Cubic”
- mix2Order – “Linear”, “Quadratic”, “Special Cubic”, “Cubic”
- noOfLackOfFit
- noOfReplicates
- noOfCenterPoints

Design information and data are specified separately from the script commands and used to initialize the program before the commands are executed, even though they appear after the script tags in the XML file. This information follows the same format used when reading or writing the design as an XML file. Alternatively, the design information can be read in from a separate file using one of the commands for loading a design file (OpenDesignFile, ImportDesign, or LoadXmlDoc).

## SetPreference

This command allows the user to set ANY preference that can be found under Edit => Preferences. It also ties into the settings for output of the evaluation, D-optimal designs, and Numerical Optimization. There are many possible settings for each preference. The following script will return DX7 to its default settings.

```
<!DOCTYPE dex7XmlDoc SYSTEM "dex7XmlDoc.DTD">
<dex7XmlDoc name=".xml" creator="DX7">
<script status="ON" logFile=".arg.log">
<!-- Math Display - General Tab -->
<scriptLine command="SetPreference" stringArg="[MathDisplay]LeverageLimit=2"/>
<scriptLine command="SetPreference" stringArg="[MathDisplay]MaxCategoricalEqns=12"/>
<scriptLine command="SetPreference" stringArg="[MathDisplay]CodedFormat=0.000"/>
<scriptLine command="SetPreference" stringArg="[MathDisplay]GeneralPrecision=6"/>
<scriptLine command="SetPreference" stringArg="[MathDisplay]ForceExponential=0"/>
<scriptLine command="SetPreference" stringArg="[MathDisplay]ForcedPrecision=3"/>
<scriptLine command="SetPreference" stringArg="[MathDisplay]OldStylePureError=0"/>
<scriptLine command="SetPreference" stringArg="[MathDisplay]SkipFactorialAliasWarning=0"/>
<!-- Math Analysis - Math Tab -->
<scriptLine command="SetPreference" stringArg="[MathAnalysis]CodedMaximum=5"/>
<scriptLine command="SetPreference" stringArg="[MathAnalysis]SignificanceThreshold=0.05"/>
<scriptLine command="SetPreference" stringArg="[MathAnalysis]AlphaStepwise=0.1"/>
<scriptLine command="SetPreference" stringArg="[MathAnalysis]CategoricSSType=2"/>
<scriptLine command="SetPreference" stringArg="[MathAnalysis]NumericSSType=3"/>
<scriptLine command="SetPreference" stringArg="[MathAnalysis]MaxOrderBeyondCubic=0"/>
<scriptLine command="SetPreference" stringArg="[MathAnalysis]DoHierarchyCheck=1"/>
<!-- not in GUI -->
<scriptLine command="SetPreference" stringArg="[MathAnalysis]ShowPseudoSimEdit=0"/>
<!-- not in GUI -->
<!-- General Graph - Graphs 1 Tab -->
<scriptLine command="SetPreference" stringArg="[GraphGeneral]GraphResolution=Medium"/>
<scriptLine command="SetPreference" stringArg="[GraphGeneral]ShowDesignPoints=1"/>
<scriptLine command="SetPreference" stringArg="[GraphGeneral]ShowGraphWarnings=1"/>
<scriptLine command="SetPreference" stringArg="[GraphGeneral]ShowGridLines=0"/>
<scriptLine command="SetPreference" stringArg="[GraphGeneral]HorizontalYLabel=0"/>
<scriptLine command="SetPreference" stringArg="[GraphGeneral]ShadingColorsUsed=3"/>
<scriptLine command="SetPreference" stringArg="[GraphGeneral]DefaultLineWidth=1"/>
<scriptLine command="SetPreference" stringArg="[GraphGeneral]DottedLineStyle=2"/>
<!-- Graph Specific - Graphs 2 Tab -->
<scriptLine command="SetPreference" stringArg="[GraphSpecific]SurfaceType3D=Graduated"/>
<scriptLine command="SetPreference"
stringArg="[GraphSpecific]ContourBackground=Graduated"/>
<scriptLine command="SetPreference" stringArg="[GraphSpecific]Vertical3DAxisOffset=25"/>
<scriptLine command="SetPreference" stringArg="[GraphSpecific]3DGraphProjectionLines=1"/>
<scriptLine command="SetPreference" stringArg="[GraphSpecific]DifferentLineStyles=1"/>
<scriptLine command="SetPreference" stringArg="[GraphSpecific]ForceYScaleAutoAdjust=1"/>
<scriptLine command="SetPreference" stringArg="[GraphSpecific]LSDDisplayType=0"/>
<scriptLine command="SetPreference" stringArg="[GraphSpecific]ParetoShowTLimits=1"/>
<scriptLine command="SetPreference" stringArg="[GraphSpecific]ParetoForceMaxTicks=1"/>
<scriptLine command="SetPreference" stringArg="[GraphSpecific]ParetoMaxTicks=31"/>
<scriptLine command="SetPreference" stringArg="[GraphSpecific]ParetoShowLenths=0"/>
<!-- not in GUI -->
<!-- Font Preferences - Fonts & Colors Tab -->
<scriptLine command="SetPreference" stringArg="[Font]SheetFont=14 Arial"/>
<scriptLine command="SetPreference" stringArg="[Font]ModelFont=14 Arial"/>
<scriptLine command="SetPreference" stringArg="[Font]ControlFont=14 Arial"/>
```

```

<scriptLine command="SetPreference" stringArg="[Font]ContourFont=18 Arial"/>
<scriptLine command="SetPreference" stringArg="[Font]LabelFont=18 Arial"/>
<scriptLine command="SetPreference" stringArg="[Font]ScaleFont=18 Arial"/>
<scriptLine command="SetPreference" stringArg="[Font]LegendFont=16 Arial"/>
<scriptLine command="SetPreference" stringArg="[Font]TitleFont=18 Arial"/>
<!-- Color Preferences - Fonts & Colors Tab -->
<scriptLine command="SetPreference" stringArg="[Color]contour=0"/>
<scriptLine command="SetPreference" stringArg="[Color]contour graph=65535"/>
<scriptLine command="SetPreference" stringArg="[Color]3d Contour Background=65535"/>
<scriptLine command="SetPreference" stringArg="[Color]hilite line=255"/>
<scriptLine command="SetPreference" stringArg="[Color]point group=33023"/>
<scriptLine command="SetPreference" stringArg="[Color]hilite point=16711680"/>
<scriptLine command="SetPreference" stringArg="[Color]design points=255"/>
<scriptLine command="SetPreference" stringArg="[Color]subsurface design points=12105983"/>
<scriptLine command="SetPreference" stringArg="[Color]point inModel=16777215"/>
<scriptLine command="SetPreference" stringArg="[Color]positive effect=33023"/>
<scriptLine command="SetPreference" stringArg="[Color]negative effect=16744448"/>
<scriptLine command="SetPreference" stringArg="[Color]BadOverlay=12632256"/>
<scriptLine command="SetPreference" stringArg="[Color]hilite text=16711680"/>
<scriptLine command="SetPreference" stringArg="[Color]high shading=255"/>
<scriptLine command="SetPreference" stringArg="[Color]low shading=16711680"/>
<scriptLine command="SetPreference" stringArg="[Color]mid shading=65280"/>
<scriptLine command="SetPreference" stringArg="[Color]grid lines=10526880"/>
<scriptLine command="SetPreference" stringArg="[Color]bargraph background=16777215"/>
<scriptLine command="SetPreference" stringArg="[Color]CatalogSurface=8454143"/>
<scriptLine command="SetPreference" stringArg="[Color]CatalogHighlight=16777215"/>
<scriptLine command="SetPreference" stringArg="[Color]CatalogShadow=4227200"/>
<!-- Evaluation Preferences -->
<scriptLine command="SetPreference" stringArg="[Evaluation]ShowEvalAliasList=1"/>
<scriptLine command="SetPreference" stringArg="[Evaluation]ShowEvalPower=1"/>
<scriptLine command="SetPreference" stringArg="[Evaluation]PowerEffectSize1=.5"/>
<scriptLine command="SetPreference" stringArg="[Evaluation]PowerEffectSize2=1"/>
<scriptLine command="SetPreference" stringArg="[Evaluation]PowerEffectSize3=2"/>
<scriptLine command="SetPreference"
stringArg="[Evaluation]PowerMaxFactorsCalculated=15"/>
<scriptLine command="SetPreference"
stringArg="[Evaluation]PowerSignificanceThreshold=2"/>
<scriptLine command="SetPreference" stringArg="[Evaluation]ShowEvalLeverage=1"/>
<scriptLine command="SetPreference" stringArg="[Evaluation]ShowEvalMatrixMeasure=0"/>
<scriptLine command="SetPreference" stringArg="[Evaluation]ShowEvalCorrelations=0"/>
<scriptLine command="SetPreference" stringArg="[Evaluation]HiliteBadCorrelations=1"/>
<scriptLine command="SetPreference" stringArg="[Evaluation]HiliteBadCorrValue=.7"/>
<scriptLine command="SetPreference"
stringArg="[Evaluation]MaxEvalCorrelationTerms=128"/>
<scriptLine command="SetPreference" stringArg="[Evaluation]UseVarianceForEvaluation=0"/>
<scriptLine command="SetPreference" stringArg="[Evaluation]ScaleStdErrByN=0"/>
<!-- Optimization Preferences -->
<scriptLine command="SetPreference" stringArg="[Optimization]OptiCycleCount=30"/>
<scriptLine command="SetPreference" stringArg="[Optimization]OptiDupSense=.5"/>
<scriptLine command="SetPreference" stringArg="[Optimization]OptiSimpFrac=.1"/>
<scriptLine command="SetPreference" stringArg="[Optimization]OptiNumDesignPoints=50"/>
<scriptLine command="SetPreference" stringArg="[Optimization]MaxOptSolns=100"/>
</script>
</dex7XmlDoc>

```

## Getting Data into Design-Expert 7 using XML

Because this document is about using an automated system with Design-Expert<sup>®</sup>, there are at least two more steps. Get the design to the automated experimentation control program, and get the data from the experiments to Design-Expert. XML format provides a standard tagging system which allows parsing of the XML document for input into the control program. In reverse, the data file created by the automated measurement process can be entered into the XML design script and then processed by Design-Expert.

We can only provide suggestions for how to accomplish these tasks. There are two choices. Either add direct XML processing capability to the automated process or create a translator to parse the XML document. The author of this document used PERL code to parse and load a flat file containing response data into an XML design file created from Design-Expert. A separate set of code is required to create a flat file to transfer the design (randomized run order, factor settings, etc) to the control program for the automated experiment.