

**swCP3: A SolidWorks Custom Properties Editor**  
**swCP3 1.54 & swCP3d 1.20**  
**April 8, 2005**

**USER MANUAL**

## **1. What is swCP3?**

swCP3 is a SolidWorks Custom Properties editor implemented as an Add-In dll. All aspects of the editor including *Custom Property Names* are fully customizable through an XML initialization file. Features of swCP3 include definition of Custom Property “Frames” and lists, checking of entered values against standard prototypes, capitalization of fields, composite fields and reading & setting SolidWorks document properties from swCP3. It is possible to enter different *Custom Property Values* for each configuration separately if so desired.

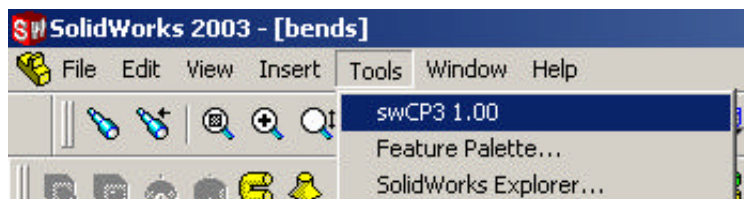
A fully customizable “get next serial number” functionality is incorporated and it can be served either locally or from a remote server<sup>1</sup>. Local operation is the default and is good if only a single user is required to consume serial numbers, while remote operation is essential if there are to be no conflicts in the use of the serial numbers among a group or users.

These custom properties may be incorporated into SolidWorks Drawing documents by entering the corresponding Custom Property Names into drawings, or into the Bill-of-Materials bringing with it the advantages of standardization of documentation across multiple users and auto-updation of filled-in information.

## **2. Installation**

Run the setup program included in the <swCP3.zip> file and follow the displayed instructions

In SolidWorks browse and “open” the swCP3.dll file as you would any other SolidWorks document. The Add-In will be installed and all required entries to the Microsoft Windows registry will be made automatically on exiting SolidWorks. The menu item for invoking <swCP3> will appear at the top of the Tools menu for either a Part or Assembly document.



## **3. Disabling the Add-In**

You can at any time disable the Add-In through the [Tools] -> [Add-Ins...] menu in SolidWorks

## **4. Uninstalling the Add-In**

You can uninstall swCP3 from the [Start]->[Settings]->[Control Panel]->[Add/Remove Programs] menu.

---

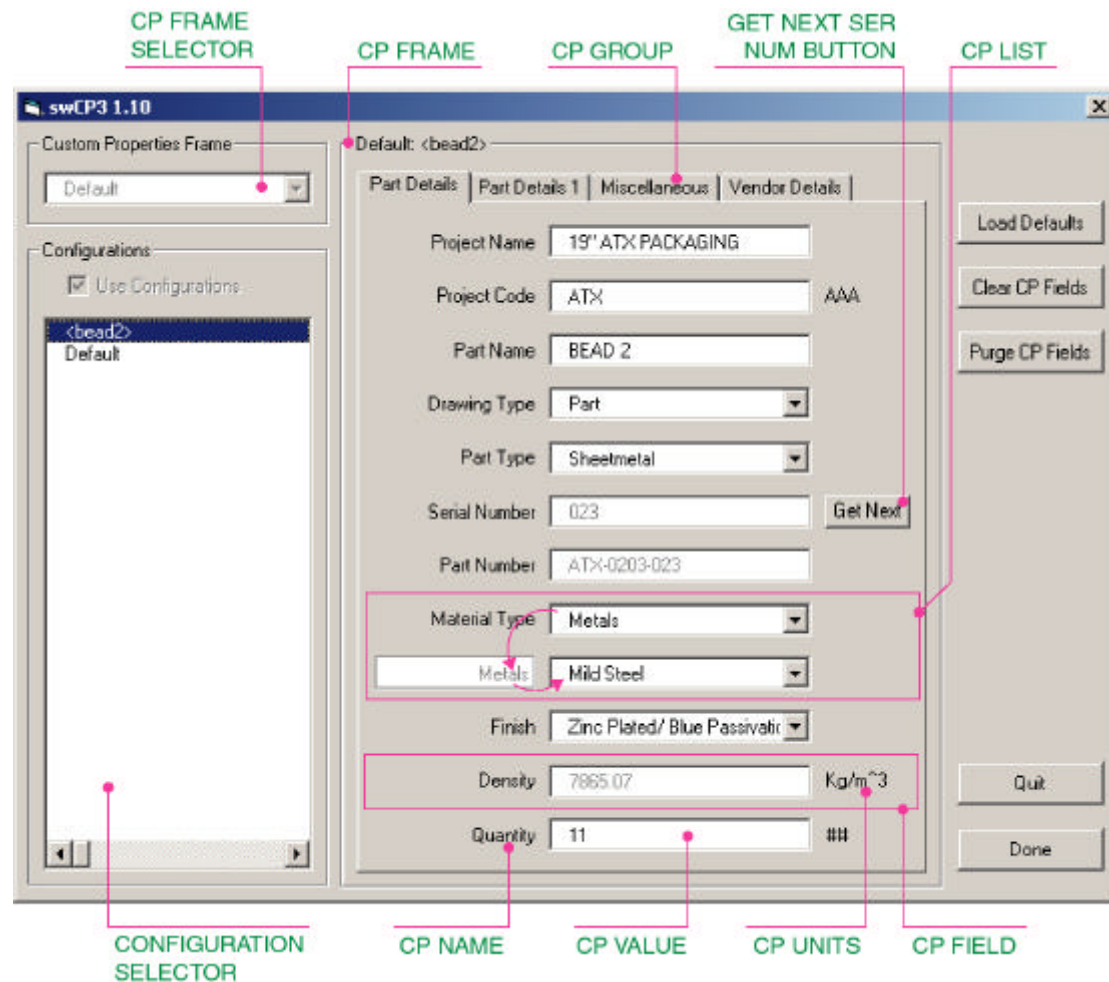
<sup>1</sup> See APPENDIX-II

## 5. Compatibility

swCP3 should work with SolidWorks 2003 and above without problems.

## 6. Operation

Invoking swCP3 from the Tools menu will display the swCP3 window constructed according to the description given in the <swCP3.xml> file.



You may enter custom property values in the various fields for the document itself or for each configuration separately by selecting the "Use Configurations" check box. Individual configuration values may be entered by selecting it in the Configuration Selector. In the first case, the values are set for the *Custom* properties and in the second case, separate values may be entered for both *Custom* and *Configuration Specific* custom properties. The following actions are taken on pressing the various buttons:

- **[Done]** causes swCP3 to save the properties for the currently selected Custom Property Frame and exit
- **[Quit]** causes swCP3 to exit without modifying the SolidWorks document in any way.

- **[Clear CP Fields]** clears all text fields and invalidates the index on list type fields in the current configuration. Right-Click to display additional options to clear CP Values from all configurations.
- **[Load Defaults]** causes default values, if present, to be loaded into *empty* CP Fields and invalidated lists only and does not affect other fields.
- **[Load Scratch]** causes previously stored CP Values to be loaded into *empty* CP Fields from the scratchpad. The values are automatically stored in the scratchpad on pressing **[Done]**. You can also save the displayed CP Values to the scratchpad or clear the scratchpad by Right-Clicking on the **[Load Scratch]** button.
- **[Copy From...]** allows you to copy CP Values from another SW document into *empty* CP Fields in the current form. You will be prompted to select the SW document file and the configuration from which you want the values to be copied.
- **[Purge CP Fields]** removes any Custom Property Names and Values from the document previously setup with swCP3.

While default values are displayed on invoking swCP3 on a fresh SolidWorks document, subsequent invocations will cause swCP3 to read and display the stored property values. The Custom Property Frame Selector and the “Use Configurations” checkbox are also disabled for any subsequent invocations of swCP3. These may be re-enabled by pressing the **[Purge CP Fields]** button, which, as mentioned above, also causes all Custom Property Names and Values to be deleted from the document.

The Custom Properties may be viewed and/ or manually edited by accessing them from the SolidWorks [File -> Properties...] menu.

## 7. Customizing swCP3

The <swCP3.xml> file controls all aspects of swCP3’s behavior. Please see<sup>2</sup> APPENDIX - I for the description of the XML file and its customization. Remember that all XML files *are* case-sensitive when editing the <swCP3.xml> file.

The file can be edited with any text editor like the Windows NOTEPAD, though that is not essentially the best method of editing an XML document; it can be a frustrating experience. A text editor with syntax highlighting can make editing easier.

However, building an XML document from scratch in a spreadsheet application like Microsoft Excel is a good idea and can prevent a lot of frustration, since you can use all of the spreadsheet’s features to build, maintain and preserve the database tree structure of an XML document. A spreadsheet is also great for visually debugging an XML document. Please see the Excel file included with this distribution for an idea on how to do this.

If you use Excel for XML editing, you need to perform the following additional steps:

- a. Select and copy the entire text block in Excel and paste into NOTEPAD. (Check that the file is empty first!)

---

<sup>2</sup> APPENDIX – III is “A Brief Introduction to XML”, for those interested

- b. Save the NOTEPAD document as an XML file (a file with extension “. xml”)
- c. Open the saved file in Internet Explorer. If the XML document is not “well-formed,” the error will be highlighted. If there are any errors, you can re-edit the Excel file and try again.

Several free and shareware XML editors are available for download. Expensive commercial XML editors are also available, and these may be used, if available.

## 8. Including Custom Property Fields into Drawing Documents

The *Custom Property Values* may be incorporated in drawing documents (either the drawing sheet itself or the drawing sheet format) using the corresponding *Custom Property Names*. It is more efficient to create a drawing template with these entries at the appropriate locations. For details, please see the SolidWorks help on how to do this.

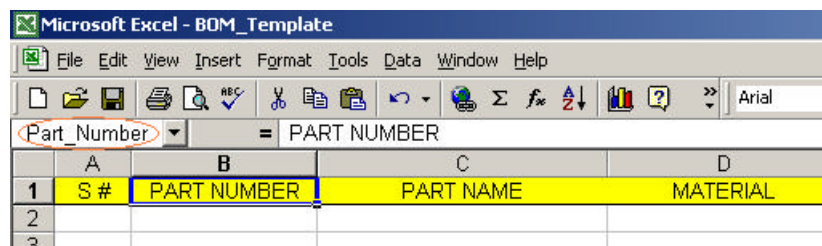
The actual value displayed in the drawing document depends on the following factors:

- a. The active view to which the annotation (field code) is “attached”
- b. The active configuration of the model in that view

Please see the SolidWorks help for more details on these aspects. I couldn’t be clearer or more confusing than the SolidWorks documentation. You may have to do a “rebuild” to update any modified data.

## 9. Including Custom Property Fields in Bill-of-Materials

You can easily incorporate these custom properties into the Bill-of-Materials by “naming” the topmost cell of the appropriate column with the *Custom Property Name* in the Bill-of-Materials template.



Please see the Microsoft Excel help for details on how to do this. Remember that certain columns in the standard SolidWorks Bill-of-Materials template are mandatory for the whole thing to work in SolidWorks. Just hide the fields you don’t want in the Excel sheet and you’ll be fine.

## 10. Additional Notes

- swCP3 will work only on Assembly and Part documents and not on Drawing documents.
- Both the swCP3.xsd and swCP3.xml files are required to be located in the same directory as the swCP3.dll file. The various field names, values and other settings and commands are read in from the XML file and validated against the description given in the Schema (swCP3.xsd) file.

- You may wish to look at the Schema file to get some idea on how the add-in works. Please do not edit this file as it may render swCP3 inoperable.
- Note that SolidWorks will copy any existing custom properties in a configuration into a newly created (dependant) configuration.
- swCP3 does not currently support SolidWorks system properties ("SW-...") and dimensional values, though, nothing prevents one from explicitly typing it out in a text field.

## 11. Limitations of swCP3

- swCP3 can handle a maximum of 9999 fields included in all "frames" and "groups". But don't try it; swCP3 may take hours to load!
- Bulk operations (save CP Values to multiple files) are not yet implemented. Check out [swC3report] at <http://www.swcp3.com/>

## 12. License Agreement

- a. swCP3 Ver 1.54 is NOT free. Please do not redistribute.
- b. I retain the copyright to everything in the software, the user interface, the customization interface and the swCP3 Schema.
- c. I retain the copyright to the *GetNextSerialNumber* functionality its operation and its implementation.
- d. This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
- e. swCP3 comes with ABSOLUTELY NO WARRANTY. See especially the following:

### NO WARRANTY

A. THERE IS NO WARRANTY FOR THE PROGRAM, TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

B. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR

LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

- f. All copyrights and trademarks are the property of their respective owners.
- g. You may wish to contact me with suggestions/ modifications to swCP3 at [info@swCP3.com](mailto:info@swCP3.com).

### 13. Revision History

<i>Version</i>	<i>Date</i>	<i>Description</i>
1.00	09-Jun-2004	<ul style="list-style-type: none"><li>• Basic version with everything working</li></ul>
1.10	14-Jun-2004	<ul style="list-style-type: none"><li>• “Get Next Serial Number” implemented</li></ul>
1.11	15-Jun-2004	<ul style="list-style-type: none"><li>• “Get Next” will now <i>not</i> work if the field has some content; it has to be cleared first</li><li>• Confirmation is now required for an index is to be generated for any new Key <i>Value</i></li><li>• Copy and paste from text and list fields</li></ul>
1.20	20-Jun-2004	<ul style="list-style-type: none"><li>• Now handles Document Summary Information</li><li>• Can handle different <i>sets</i> of Custom Property Fields for the document and for the configurations</li><li>• “settings” section added to schema for future extensions</li></ul>
1.30/ 1.00	22-Jun-2004	<ul style="list-style-type: none"><li>• swCP3d Ver 1.00 up and running, serving and maintaining the indices on a remote server.</li></ul>
1.31/ 1.11	23-Jun-2004	<ul style="list-style-type: none"><li>• Network related errors now handled gracefully.</li><li>• swCP3d uses ebCrypt library by Alexei Bakharevski [<a href="mailto:ebcrypt@ihug.com.au">ebcrypt@ihug.com.au</a>] for license management</li></ul>
1.32/ 1.20	18-Jul-2004	<ul style="list-style-type: none"><li>• swCP3d uses Blowfish encryption code by David Ireland of DI Management Services Pty Ltd. [<a href="http://www.di-mgt.com.au">www.di-mgt.com.au</a>] instead of the ebCrypt library</li></ul>
1.40/ 1.20	15-Nov-2004	<ul style="list-style-type: none"><li>• Implemented [Load Scratch] to load previously stored CP Values from registry</li><li>• Implemented [Copy From...] to load CP Values from another SW Document</li></ul>
1.41/ 1.20	19-Nov-2004	<ul style="list-style-type: none"><li>• Finally fixed the “two mouse clicks required” problem</li></ul>
1.44/ 1.20	13-Mar-2005	<ul style="list-style-type: none"><li>• Local editing of serial numbers w/o modifying them in the registry</li></ul>
1.52/ 1.20	21-Mar-2005	<ul style="list-style-type: none"><li>• Added minVal, maxVal and incVal for “GetNextSerialNumber” Fields</li><li>• systemWrite for “filename”, “setTitle” and “units” implemented</li></ul>
1.54/ 1.20	08-Apr-2005	<ul style="list-style-type: none"><li>• “GetNextSerialNumber” works only when there is a key value present, adding yet another bug in the process...</li></ul>



## APPENDIX – I

### Customizing swCP3

The <swCP3.xml> file and the Schema <swCP3.xsd> must be located in the same directory as the <swCP3.dll> file. The XML document controls all aspects of the behavior of swCP3 within the context of its operation in SolidWorks. The following features of swCP3 are customizable:

- Custom Property Field Names
- Custom Property Frames. Frames hold distinct and different collections of Custom Property Fields. In actual usage, one would choose only one “Frame” that is applicable to a particular SolidWorks document
- Custom Property Groups. Groups are a means of conveniently organizing Custom Property fields. They have no other function in swCP3.
- Custom Property Lists. These are lists of Custom Property *Names* from which the applicable one may be selected. Think of it as a “Frame” (described above), but applicable to a single CP Field.
- Custom Property Value attributes such as capitalization, if a field is editable, prototypes, field lengths or default values. It has a fully customizable “get next serial number” functionality, including “local” and “remote” operation.
- “Composite” fields. Composite fields in swCP3 are any combination of other Custom Property Values and text strings
- Reading and setting of SolidWorks document settings and values (like filename or density) from swCP3.

Please remember that the entire point in using any Custom Properties editor such as this one is to standardize Custom Property “Name” strings and impose restrictions on Custom Property “Values” entered by users.

#### 1. The swCP3 <root> Element

For reasons that are too messy to go into, the following is required to be present in the <swCP3.xml> file for anything to happen in swCP3 (my comments are in **orange**):

---

```
<?xml version = "1.0" encoding="UTF-8"?>
<!-- swCP3.xml,
      Copyright (C) 2004 Vinodh Kumar M., 07-Apr-2004 -->

<root xmlns="http://www.swCP3.com/"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.swCP3.com/ swCP3.xsd">
  (ALL CONTENT GOES HERE)
  (THERE IS A SPACE HERE)
</root>
```

---



Only *one* <root> element is permissible in the <swCP3.xml> file. This means that everything must be located in the region between the <root> *start* and *end* tags as shown above. Don't bother with what look like WWW site-names, they only *look* like URLs, and their purpose here is entirely different!

The second line between “<!--” and “-->” is a comment that can be modified or deleted if you so wish.

If you copy and paste the above into NOTEPAD and save it as <swCP3.xml> in the proper directory and then try to load swCP3, you will start getting meaningful error messages.

## 2. Custom Property Frames: The <frame> Element

You can think of Custom Property Frames (CP Frames) as distinct drawing annotation schemes. In practice, one would select and use only one applicable scheme for a particular document. This can be useful if you use different part numbering schemes for different types of parts, or if you have several clients, each having their own drawing annotation requirements. No relationships can exist between frames, either implied or inferred.

You can include up to 25 frames in swCP3 by including them within the <root> element in the swCP3.xml document. The following shows the <root> element containing three <frame> elements:

---

```
<?xml version = "1.0" encoding="UTF-8"?>
<!-- swCP3.xml,
      Copyright (C) 2004 Vinodh Kumar M., 07-Apr-2004 -->

<root xmlns="http://www.swCP3.com/"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.swCP3.com/ swCP3.xsd">

    <frame name="Frame One">
        (FRAME ONE CONTENT)
    </frame>

    <frame name="Frame Two and a Half">
        (FRAME TWO CONTENT)
    </frame>

    <frame name="You can give any name">
        (FRAME THREE CONTENT)
    </frame>

</root>
```

---

The following rules apply to <frame> elements:

- The <root> element can contain *only* <frame> elements, no other elements are permissible (except for the optional <settings> element, described in a later section. Don't bother with it now.)
- The <frame> element can appear only within the <root> element

- Nothing can appear between <frame> elements
- The “name” attribute is compulsory, and any valid string value can be used

### 3. Custom Property Groups: The <group> Element

Custom Property Groups (CP Groups) are convenient means of organizing (and grouping) Custom Property Fields and presenting them to the user. They do not have any other function in swCP3. Up to 25 <group> elements can be included within each <frame>. The following shows four <group> elements within a <frame> element.

---

```
<?xml version = "1.0" encoding="UTF-8"?>
<!-- swCP3.xml,
      Copyright (C) 2004  Vinodh Kumar M., 07-Apr-2004 -->

<root xmlns="http://www.swCP3.com/"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.swCP3.com/ swCP3.xsd">

  <frame name="Frame One">

    <group name="Group one">
      (GROUP ONE CONTENT)
    </group>

    <group name="Vendor Details">
      (GROUP TWO CONTENT)
    </group>

    <group name="GRP II">
      (GROUP THREE CONTENT)
    </group>

    <group name="Steam">
      (GROUP FOUR CONTENT)
    </group>

  </frame>
  (ONLY ONE FRAME IS SHOWN, OTHER FRAMES MAY BE PRESENT)
</root>
```

---

The following rules apply to <group> elements:

- The <frame> element can contain *only* <group> elements, no other elements are permissible
- The <group> element can appear only within the <frame> element, you cannot put it inside the <root> element, for example.
- Nothing can appear between <group> elements
- The “name” attribute is compulsory, and any valid string value can be used

#### 4. Custom Property Fields: The <custProp> Element

Each Custom Property Field (CP Field) holds the actual Custom Property *Name* and *Value* that is entered into document properties on pressing the “Done” button in swCP3. Up to 200 CP Fields may be included in each group (but don’t try it!). The following shows two CP Fields included in a <group> located in a <frame>:

---

```
<?xml version = "1.0" encoding="UTF-8"?>
<!-- swCP3.xml,
      Copyright (C) 2004 Vinodh Kumar M., 07-Apr-2004 -->

<root xmlns="http://www.swCP3.com/"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://www.swCP3.com/ swCP3.xsd">

  <frame name="PART NUMBERING SCHEME">
    <group name="PURCHASE INFORMATION">

      <custProp CPTag="Vendor_Name">
        (OTHER ELEMENTS MAY BE PRESENT HERE
         AND ARE DISCUSSED BELOW)
      </custProp>

      <custProp CPTag="Vendor_Part_Num">
        (OTHER ELEMENTS MAY BE PRESENT HERE
         AND ARE DISCUSSED BELOW)
      </custProp>

    </group>
    (ONLY ONE GROUP IS SHOWN, OTHER GROUPS MAY EXIST)
  </frame>
  (ONLY ONE FRAME IS SHOWN, OTHER FRAMES MAY BE PRESENT)
</root>
```

---

The following rules apply to <custProp> elements:

- The <custProp> element can appear only within the <group> element and not outside of it.
- The “CPTag” attribute is compulsory, and any valid string value can be used.

The above XML fragment – without the **comments** – is the bare minimum necessary for properly constructing and displaying the swCP3 form without errors. At this stage, if there are no spelling or syntactical mistakes, the swCP3 will successfully load and display one CP Frame containing one CP Group containing two CP Fields.

If you used the fragment above, you will notice that the “CPTag” attribute *values* (“Vendor\_Name” and “Vendor\_Part\_Num”) are displayed as labels alongside empty textboxes. Anything typed into these CP Value textboxes are saved in the Document Custom Properties with the CPTag strings as *Custom Property Names* and the typed-in values as corresponding *Custom Property Values*. These may be viewed and edited from the SolidWorks [File -> Properties...] menu.

As mentioned earlier, the <frame> and <group> elements are used merely for organizing CP Fields and there is nothing more to them. The CP Field's behaviour however, is controlled through the several attributes associated with the <custProp> element (See the swCP3 window image on page 2.) Here is the complete list and description of all <custProp> attributes:

**a. <custProp> Attribute: CPTag**

Type	String
Bounds	
Default value	
Remarks	Compulsory attribute

We have just seen above how the compulsory attribute CPTag is used. It holds the string that is used as the Custom Property *Name*, and in the absence of the "displayName" attribute (See b., below) in the <custProp> element, also used as the label value for CP Name and displayed in front of CP Value in a CP Field.

**Useful tip:** It would be preferable to use strings without spaces or tabs in them for CPTag strings for compatibility with other software. It is also essential that *unique* strings be chosen for each of the CPTags within a frame to prevent unnecessary confusion. However some setups may necessitate use of common CPTags.

**b. <custProp> Attribute: displayName**

Type	String
Bounds	
Default value	<empty>
Remarks	First 16 characters displayed

The value of this attribute, if present, is used as the label value for CP Name and displayed in front of the CP Value in a CP Field. Its value may be as long as you want but only the first 16 characters are displayed. Don't confuse "displayName" with "CPTag", which is always used as the value for Custom Property *Name*.

**c. <custProp> Attribute: units**

Type	String
Bounds	
Default value	<empty>
Remarks	First 6 characters displayed

The value of this attribute, if present, is used as a label value for CP Units and displayed alongside CP Value in a CP Field. The string value may be as long as you want but only the first 8 characters are displayed. This may be used to present optional, additional information to the user. It has no other function in swCP3.

**d. <custProp> Attribute: length**

Type	Integer
Bounds	"1" to "100"
Default value	"100"
Remarks	Applies to textbox only

This is an integer value between 1 to 100 that sets the length of a CP Value textbox. Strings longer than “length” cannot be typed in.

**e. <custProp> Attribute: allCaps**

Type	Boolean
Bounds	“true” or “false”
Default value	“false”
Remarks	Applies to textbox only

If set to “true”, causes characters typed into the CP Value textbox to be capitalized.

**f. <custProp> Attribute: editable**

Type	Boolean
Bounds	“true” or “false”
Default value	“true”
Remarks	

If set to “false”, the user is prevented from modifying the corresponding CP Values.

**g. <custProp> Attribute: sameForAllConfigs**

Type	Boolean
Bounds	“true” or “false”
Default value	“false”
Remarks	

If set to “true”, the users are prevented from entering different CP Values for different configurations. The same value is used for all Configurations and for the Document Custom Property. This is useful for common annotations like Project Names and such.

**h. <custProp> Attribute: location**

Type	String
Bounds	“onlyDocument”, “onlyConfigurations & “bothDocumentAndConfigurations”
Default value	“bothDocumentAndConfigurations”
Remarks	

This attribute is used to specify if the CP Field is to be displayed for the Document Custom Properties, Configuration Specific Custom Properties or for both.

Though any “location” value can be used with “standard” and “list,” there are some limitations of using “location” in combination with other types. You must, for example, ensure that CP Lists and “composite” fields, when used together, lie in the same “location.” Also, some fields are really valid only with “onlyDocument” (e.g. Document Summary Information or systemWrite).

**Caution:** Several checks have been incorporated to prevent building an unusable swCP3 form, but there is no way of knowing if every input combination has been accounted for. Be careful when using “location” with CP Lists and “composite” fields.

**i. <custProp> Attribute: prototype**

Type	String
Bounds	Any character string
Default value	
Remarks	Applies to textbox only

This attribute provides a *pattern* string (a “prototype”) that the entered CP Value should look like. The entered CP Value is compared with the pattern and only values that match are allowed. This is useful to ensure that users enter similar CP Values for fields like version numbers and part codes.

Be careful when using this with other attribute settings like “allCaps” and “length”, you may create a deadlock situation. It is best to use only the prototype value when in doubt.

The prototype string can be composed of the following character patterns:

<i>Character</i>	<i>Matches</i>	<i>Example</i>
?	Any <i>one</i> character	“?” will match “A” “a” or “z” “?-??” will match “A-kR”
*	Zero or more characters	“a*a” will match “aa” and “aGSa” “a??*a” will match both “adfa” and “apRtasdfasFa”
#	Any digit 0-9	“###” will match “156” “#.##.#” will match “1.34.1” “?#-##” will match “B1-34”
[charlist]	Any character in charlist	“[abc]” will match “a” or “b” or “c” “[cpx]” will match “c” or “p” or “x”
[!charlist]	Any character other than those in charlist	“[!abc]” will match “d” thru “z”
Any other character	Itself	“?tests?#ING#” will match “XtestsT4ING5”

One can build arbitrarily complex patterns to require the user to type in only those strings that match. In swCP3, a blank CP Value will always match; i.e. the user may choose to leave the CP Value blank.

**j. <custProp> Attribute: type**

Type	TOKEN
Bounds	“standard”, “list”, “composite” or “systemRead”
Default value	“standard”
Remarks	

The “type” attribute controls a CP Field’s behavioral characteristics. These are listed below:

<i>Attribute value</i>	<i>Effect</i>
“standard”	This is the default value of “type” attribute. A textbox is displayed and the user may freely type-in any string subject to any limitations imposed by other attributes (namely “allCaps”, “length” and “prototype”)

	See section on “<var> elements and “standard” CP Fields” on how to specify default CP Values for this type.
“list”	A listbox is displayed and the user is limited to any selection within the displayed list.  See section on “<var> elements and “list” CP Fields” on how to specify lists and default values for lists.
“composite”	This is a combination of other CP Values and may be editable or not.  Subsequent behavior of this field is dependant on the value of the “editable” attribute. If it is <i>not</i> editable, the value is rebuilt every time something changes in the constituent CP Values. If it is editable, the value is not rebuilt and does not reflect changes in the constituent CP Values.  See section on “<var> elements and “composite” CP Fields” on how to specify a composite field.
“systemRead”	The specified system value or SolidWorks document settings is read into the CP Value during swCP3 start-up if the CP Value is <i>not</i> editable, or read from the Document Custom Property if it is editable.  See section on “<var> elements and “systemRead” CP Fields” on how to specify what system value or SW document setting is read into the CP field.

**k. <custProp> Attribute: systemWrite**

Type	Boolean
Bounds	“true” or “false”
Default value	“false”
Remarks	

If set to “true”, enables the CP Value to be written into the SolidWorks document settings determined by the value of “systemWriteValue”(See: l, below)

**l. <custProp> Attribute: systemWriteValue**

Type	String
Bounds	“density”, “summaryTitle”, “summaryAuthor”, “summaryKeywords”, “summaryComments”, “summarySubject”, “filename”, “setTitle”, “units”
Default value	<empty>
Remarks	other settings not implemented

The string determines the document setting (e.g. “density”, “filename”, “units” etc) the CP Value is written into. Use this attribute to save values in Document Summary Information. **Note:** “filename” will rename *and* save the file to the currently selected



SolidWorks directory while “setTitle” will only set the filename in a *new* SW document. **Important:** With “setTitle”, files already saved to disk will not be renamed.

**m. <custProp> Attribute: minVal**

Type	String
Bounds	
Default value	“0”
Remarks	

This attribute is used to specify the minimum value for the GetNextSerialNumber field. If specified along with any “prototype” attribute value, it must match the prototype specification (do not escape any literal “#” or “?” values as you would in the “prototype” attribute). “minVal” has no significance if used for any other field other than for “GetNextSerialNumber”.

**n. <custProp> Attribute: maxVal**

Type	String
Bounds	
Default value	(max allowed for “long” value)
Remarks	

This attribute is used to specify the maximum value for the GetNextSerialNumber field. If specified along with any “prototype” attribute value, it must match the prototype specification (do not escape any literal “#” or “?” values as you would in the “prototype” attribute). “maxVal” has no significance if used for any other field other than for “GetNextSerialNumber”.

**o. <custProp> Attribute: incVal**

Type	Integer
Bounds	
Default value	1
Remarks	

“incVal” is used to specify the incremental step value for the GetNextSerialNumber functionality. “incVal” has no significance if used for any other field other than for “GetNextSerialNumber”.

**p. <custProp> Attribute: reportPrintColumn**

Type	Boolean
Bounds	“true” or “false”
Default value	“true”
Remarks	

This attribute is used only by swCP3report and is used to specify if a CP Value column is to be printed or saved. This property is overridable in swCP3report.

**q. <custProp> Attribute: reportTextAlignment**

Type	String
------	--------

Bounds	"left", "center", "right"
Default value	"left"
Remarks	

This is used to specify the CP column text alignment in swCP3report.

You can use any or all of the attribute tags (once, in any order, within the start <custProp> tag) as shown below (the <root>, <frame> and <group> elements are not shown):

---

```
<custProp
    CPTag="Project_Code"
    type="standard"
    displayName="Project Code"
    units="A##"
    sameForAllConfigs="true"
    location="onlyDocument"
    prototype="[A-Z][A-Z0-9][A-Z0-9]"
    reportTextAlignment="left">

</custProp>
```

---

## 5. The Values Database: The <var> Element

Each <custProp> element can contain any number of <var> elements. The <var> element is used to hold content that is used to fill CP Values. How this is done depends on the settings of the "type" attribute in the <custProp> element.

The <var> element, if present within a <custProp> element, *must* also contain the <x> element. The <y> and <z> elements are optional, but if present, must be physically located in the same order (<x><x>, <y></y>, <z></z>).

You can arbitrarily assign any value and meaning to the <x>, <y> and <z> elements as long as the meanings are consistent within a <custProp> element. In the following example, the <x> element holds the Initials of a person, the <y> element holds the department name and the <z> element holds the person's telephone number:

---

```
<custProp CPTag="Designed_By" type="list">

    <var>
        <x> SVK </x>
        <y> Design Department </y>
        <z> 482 </z>
    </var>

    <var>
        <x> MVK </x>
        <y> Design Department </y>
        <z> 231 </z>
    </var>
```

---

(THOUGH ESSENTIALLY THE SAME AS THOSE ABOVE, SOME

---

MAY FIND THE FOLLOWING FORMAT EASY TO VISUALISE AND EDIT)

```
<var> <x>LCP</x> <y>Prototyping</y> <z>442</z> </var>
<var> <x>PYA</x> <y>Design Dept.</y> <z>462</z> </var>

</custProp>
```

---

The following rules apply to <var> elements:

- The <var> element can appear only within the <custProp> element and nowhere else.
- The <var> element *must* contain at least one <x> element, the <y> and <z> elements are optional.
- If the <y> and <z> elements are present they must be in the ascending order (<x><x>, <y></y>, <z></z>).
- No other elements can be located between successive <var> elements and between the <x>, <y> and <z> elements.

**Useful Tip:** Though any number of <var> elements may be present within each <custProp>, in the interests of speed of loading of swCP3, avoid including more than 25 or so in each of them, at least till you get familiar with swCP3's behavior.

#### a. <var> elements and “standard” CP Fields

With “standard” CP Fields, the default values are stored in the <x> element. Only the (compulsory) <x> element located within the <var> element has significance.

As it is, only the first <var> element is used as the default value. You can override it by using the “default” attribute value in the <var> element. The “default” attribute can take the values “true” or “false”. You can specify any number of default <var> elements but the last one is used. An example:

---

```
<custProp CPTag="User" displayName="User Initials"
  type="standard">

  <var>
    <x>MVK</x> <-- NORMALLY, THIS WOULD
  </var>                                BE THE DEFAULT VALUE

  <var><x>SVK</x></var>

  <var default="true">
    <x>LCP</x> <-- BUT THIS IS USED INSTEAD
  </var>

  <var><x>STR</x></var>
</custProp>
```

---

## b. <var> elements and “list” CP Fields

In the case of “list” type fields the definition remains exactly the same as the “standard” one above. Only the user will be presented with a list with all the <var> elements as choices in a list selection box. The <y> and <z> values may assume significance depending on how the CP Value is used. In any case, only the <x> value is written into Custom Properties.

## c. <var> elements and “composite” CP Fields

Any number of other CP Values (within a frame) can be included into a “composite” CP Field. It is easy to do this by including the “CPTag” value of each element to be included into a <x> element in the “composite” CP Field. If the CPTag does not exist (within the frame) the <x> value is copied without modification into the composite CP Value. Do not try to use a “composite” CP Field within itself, the application may crash. Here is an example with explanations:

---

```
<custProp CPTag="Prj_Code" type="standard"></custProp>

<custProp CPTag="Ser_Num" type="standard"></custProp>

<custProp CPTag="Order_Code" type="composite">

    <var> <x>CFOC-</x>    </var>

    <var> <x>Prj_Code</x> </var>

    <var> <x>--</x>        </var>

    <var> <x>Ser_Num</x>   </var>

</custProp>
```

IN THE ABOVE, IF Prj\_Code = "ATX" AND Ser\_Num = "034",  
THEN Order\_Code IS SET TO "CFOC-ATX-034"

---

If list values are included into a composite field, you may select which <var> element you want to include into the composite value with the <var> “useValue” attribute. Here we go:

---

```
<custProp CPTag="Drawing_Type" type="list">
    <var> <x>Assembly</x> <y>01</y> </var>
    <var> <x>Part</x>      <y>02</y> </var>
    <var> <x>Other</x>     <y>00</y> </var>
</custProp>

<custProp CPTag="Prj_Code" type="standard"></custProp>

<custProp CPTag="Prt_Num" type="composite">

    <var>                <x>Prj_Code</x>        </var>
    <var>                <x>--</x>                </var>
    <var useValue="y"> <x>Drawing_Type</x> </var>
```

---

---

```
</custProp>
```

```
IF Prj_Code = "NCP" AND Drawing_Type = "Assembly"  
THEN Prt_Num IS SET TO "NCP-01"
```

---

#### d. <var> elements and "systemRead" CP Fields

The following settings can be read from a SolidWorks document and written into the CP Value field:

<i>String</i>	<i>Value example</i>
"systemDate"	09-Jun-2004 15:23:03
"systemTime"	15:23:03
"systemShortDate"	09-Jun-2004
"systemLongDate"	Jun 09, 2004
"dateCreated"	9-Jun-2004 15:23:03 Date the SW doc was created
"dateModified"	9-Jun-2004 15:23:03 Date the SW doc was last modified
"dateAccessed"	9-Jun-2004 15:23:03 Date the SW doc was last accessed
"fileSize"	119.23 Kb File size in Kb
"fileName"	SW "title" string, see SW documentation
"fileNameWithPath"	The SW doc's filename with full path
"swDocType"	The SW doc type; "Part" or "Assembly"
"mass"	The mass as reported by SW in system units (SI)
"volume"	The volume as reported by SW in system units (SI)
"area"	The surface area as reported by SW in system units (SI)
"density"	The density as reported by SW in system units (SI)
"nextSerialNumber"	See details below

This is done by including the string corresponding to the required value inside the <x> element like this:

---

```
(THE CURRENT TIME IS COPIED TO THE "time" CP VALUE)
```

```
<custProp CPTag="time" type="systemRead">
```

```
  <var> <x>systemTime</x> </var>
```

```
</custProp>
```

```
(THE AREA IS COPIED TO THE "Component_Area" CP VALUE)
```

```
<custProp CPTag="Component_Area" type="systemRead">
```

```
  <var> <x>area</x> </var>
```

```
</custProp>
```

---

**Important:** Don't confuse "systemRead" with "systemWrite"; "systemRead" is an attribute *value* of the "type" attribute while "systemWrite" is itself a (Boolean) attribute that can be set to "true" or "false". They are used differently though they look similar.

The "nextSerialNumber" functionality is implemented differently than the other "systemRead" functions. Like the others, "nextSerialNumber" must appear as the <x> element. There are two options that control its behaviour; the "prototype" tag in the <custProp> element and the <y> element.

The optional "prototype" string, if specified, formats the output value. Its behavior is different than as described earlier. In this case, a "#" character is a placeholder for a numerical digit and a "?" is a placeholder for an *uppercase* character from "A" to "Z". Any other characters are simply copied to the output. A "#" or "?" can be "escaped" with a "\" character. This means that the required part number format must be fully described in the prototype. If no prototype is specified, then, a non-formatted running serial number is output. The retrieved number will start over from zero on reaching the maximum possible as specified in the prototype.

Also, optionally, the <y> element is used to locate a "key" *value* that is used to keep track of which index is retrieved. This gives users the flexibility to obtain the next number based on any key they choose, for example, the part type or project name or maybe the material used. This is done by setting the required "CPTag" (which is the "key") as the <y> element. Its current CP Value (which is the key *value*) will be used to retrieve the index. If no <y> element is specified, or if the "key" itself is not locatable, then, the index is retrieved from a common pool. If an unavailable key value is specified, then a new index is created for it, starting from zero. It is possible to have any number of keys and indices in a CP Frame. There are also no restrictions on the usage of the generated "nextSerialNumber" value; you can use it in a composite field, for example.

Here is an example, where the numbering is done project wise:

---

```
<custProp CPTag="Project_Code"> <-----+
</custProp>

<custProp CPTag="Part_Number"
  displayName="Part Number"
  prototype="XYZ Corp. \# ?#-###"
  type="systemRead">
  <var>
    <x>nextSerialNumber</x>
    <y>Project_Code</y> <-----+
  </var>
</custProp>
```

SPECIFYING  
A "KEY"

HERE, "Part\_Number" COULD BE "XYZ Corp. # R4-456"

---

In the above, anytime the same "Project\_Code" *value* is entered in a document, (before the "Get Next" button is pressed) the next number retrieved is the first available index held by that value.

The "Get Next" button is disabled (the caption is displayed struck-out – "~~Get Next~~") once it is pressed and any subsequent presses of the button will not cause the value

to change. Re-enabling the button requires the value to be cleared by right-clicking on the button and selecting “Clear.”

The retrieved value displayed in the “Set Serial Number” textbox is the maximum currently available free index retrieved from the relevant pool and in conformance with any specified “minVal”, “maxVal” and “incVal” attribute values.

Right-Clicking the “Get Next” button also displays other options for either zeroing the index, pre-setting it to a particular starting value for the corresponding key value and editing the serial number value locally. If the serial number is edited locally, then it is *not* checked for conformance with “minVal”, “maxVal” and “incVal” attribute values.

**Important:** The index itself is stored as a “long” type. Only the output is formatted as specified by the prototype string. This permits the index value to be formatted in any number of different ways within a frame or across documents. This may be good, or not, depending on the viewpoint.

## 6. Custom Property Lists: The <custPropList> Element

A Custom Property List (CP List) is a set of CP Fields from which only one CP Field is used in a particular document or configuration. This is similar to the CP Frame discussed earlier, but the selections are now made among a set of CP Fields included in the CP List.

Here is an example of a CP List having two CP Fields (the <root>, <frame> and <group> elements are not shown):

---

```
<custProp CPTag="Material_Type">          <-----+
</custProp>                                |
<custPropList linkTo="Material_Type">      <-----+  <- LINKED
    <custProp CPTag="Metals">
    </custProp>
    <custProp CPTag="Plastics">
    </custProp>
</custPropList>
```

---

The following rules apply:

- The <custPropList> element can appear only within the <group> element and not outside of it (or anywhere else).
- The “linkTo” attribute is compulsory, and *must* match the tag of a “CPTag” attribute of a <custProp> element located *outside* it.
- Only <custProp> elements may appear inside the <custPropList> element.



- A minimum of two <custProp> elements within a <custPropList> element are compulsory.
- Only one level of <custPropList> is allowed, and these cannot be nested.

Each <custProp> element located inside a <custPropList> element can have its own set of attributes and values and no restrictions are imposed on them. In-fact even the “CPTag” attribute may not be unique, and in many cases you may want them to be the identical.

In case the “CPTag” attributes are required to be identical, you can specify different “displayName” attributes to distinguish between the CP List elements, like this:

---

```
<custPropList linkTo="Material_Type">

    <custProp CPTag="Material" displayName="Metals">

    </custProp>

    <custProp CPTag="Material" displayName="Plastics">

    </custProp>

</custPropList>
```

---

## 7. Controlling swCP3 operation: The <settings> element

The optional <settings> element includes several attributes that control specific facets of swCP3’s behaviour.

The following rules apply to the <settings> element:

- The <settings> element can appear only within the <root> element.
- Only one <settings> element is permissible in the <root> element.
- The <settings> element, if present, must be the first element in the <root> element, ahead of any <frame> elements.

### a. <settings> Attribute: defaultFrame

Type	Integer
Bounds	0 to 25
Default value	0
Remarks	

This is used to specify the default frame that is displayed on invoking swCP3 in a new SolidWorks document.

### b. <settings> Attribute: useConfigs

Type	Boolean
Bounds	“true”, “false”
Default value	“false”
Remarks	

This determines whether the “Use Configurations” checkbox is set or not on invoking swCP3 in a new SolidWorks document.

**c. <settings> Attribute: operation**

Type	String
Bounds	“local”, “remote”
Default value	“local”
Remarks	

This determines if the “get next serial number” is to be retrieved from a remote server. The default, “local,” determines that all serial number indices are stored on and retrieved from the local machine.

**d. <settings> Attribute: serverName**

Type	String
Bounds	
Default value	
Remarks	

This is the IP address of the server that is running the “get next serial number” server <swCP3d><sup>3</sup>.

**e. <settings> Attribute: serverPort**

Type	String
Bounds	
Default value	39999
Remarks	

The port used to connect to the “get next serial number” server. This is specified when swCP3d is started on the remote server. You must confirm which port the server is listening to and set the appropriate value to enable correct operation.

Any “serverName” and “serverPort” values are ignored if operation is set to “local”

---

<sup>3</sup> See APPENDIX-II

## APPENDIX-II

### swCP3d: The “Get Next Serial Number” Server and License Manager Version 1.20

swCP3d is a simple TCP server<sup>4</sup> that keeps track of all keys, key values, and indices and serves any incoming requests from swCP3 clients for “next serial numbers.” over a LAN network.

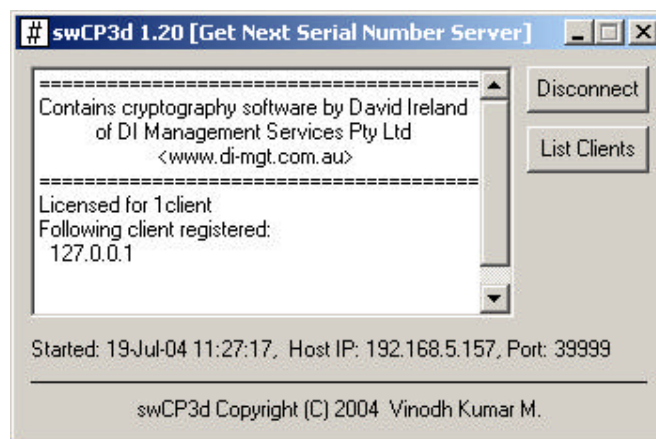
To install, copy the <swCP3d.exe> file into some convenient directory on a computer designated (*just* pick any one of the bunch...) as the “remote server”<sup>5</sup> and start-it up by double-clicking on it.

If it is the first time, you will be asked for the license key to activate the license. Copy and paste the license key<sup>6</sup> and press [GO]. This is a one-time operation.

A window for inputting the port number will be displayed<sup>7</sup>. You can select any port number you wish (if you know what you are doing), but if you are not too sure, just go with the default.

If you run into network problems,<sup>8</sup> don’t panic! Try the following: 1) [The brute-force method that I favor] – Simply exit <swCP3d> and try with a different port number. Always use a number greater than 5000, though. 2) [The scientific method] – Consult your system administrator and request a port number that is not used by any other services in your network.

The swCP3d window will be displayed on pressing [Run]. You can minimize the window and it will keep working in the background while its icon (#) will be displayed in the system tray.



The number of clients served is determined by the license purchased for the swCP3 installation. The clients (swCP3) automatically register their IP Addresses with swCP3d when they are invoked for the *first* time from SolidWorks. Once the number of licenses are exhausted, subsequent requests will be honored only from registered clients and requests from other clients will be denied.

<sup>4</sup> Only a Microsoft Windows version is available as of Version 1.43/ 1.20

<sup>5</sup> Of-course, a client can be on the same machine as the server if you so wish

<sup>6</sup> This key is included in the email sent to you on completing your purchase transaction.

<sup>7</sup> If you set the “swCP3\_PORT” Environment Variable to a valid port number, this dialog will *not* be displayed. This can be useful if you want to run swCP3d as a service and start it automatically at boot time. See the Microsoft Windows documentation on how to do this.

<sup>8</sup> The worst that can happen is swCP3d may not work as expected...

The [\[Disconnect\]](#) button is used to disconnect clients that have not terminated a connection gracefully thus resulting in a lock-up. This may sometimes happen in case certain sequence of operations have not been transacted fully by the client. The log that is displayed may be useful in troubleshooting on occasion. The [\[List Clients\]](#) button can be used to list all the registered client IP Addresses.

Remember to communicate the “remote server” IP address and the port number to all your users who will have to incorporate these settings into their swCP3.xml files.

**Important:** You can run only one *instance* of swCP3d bound to a particular port on any computer. You can however run swCP3d on as many computers as you like – with each serving a set of users – but the servers will not work collaboratively, i.e. each one will maintain it's own keys, key values and indices and there is no mechanism for synchronizing them.

## APPENDIX – III

### A Brief Introduction to XML<sup>9</sup>

This is a brief introduction on XML for people who have no clue about what it is. It should familiarize you with XML terminology and enable you to edit the <swCP3.xml> file to suit your specific needs. Please refer to the vast resources available on XML online for complete details and technically accurate descriptions.

#### 1. Elements

An XML document is a plain text document where the content is “tagged” or “marked-up”<sup>10</sup> to describe it (the content). This is done by enclosing the content between a start and an end tag – hence tags always come in pairs – and the entire construct is called an *element*. Tags are text strings placed between “<” (less than) and “>” (greater than) characters. An end tag will have the same string as the start tag but with a “/” (slash) character attached in front of it.

Here are two examples of elements:

---

```
<planet>Earth</planet>
```

IN THE ABOVE, “planet” IS A TAG, “Earth” IS THE CONTENT AND THE WHOLE THING IS AN ELEMENT...

```
<Surprise>He was surprised, he said.</Surprise>
```

---

XML is case-sensitive. The cases of start and end tags must match:

---

THE FOLLOWING WILL WORK...

```
<plaNet>Earth</plaNet>
```

... WHILE THIS WILL NOT

```
<pLaNeT>Earth</Planet>
```

---

Fundamental to everything XML is the rule that elements can be nested one (or several) inside another, like this:

---

```
<planet> <-----+
  <satellite> <----+
    Moon      |
  </satellite> <----+
  <satellite> <-----+
    Sputnik   |
  </satellite> <----+
  <rings>      <-----+
  </rings>     <----+
</planet> <-----+
```

PROPERLY NESTED ELEMENTS

---

<sup>9</sup> XML stands for “Extensible Markup Language”

<sup>10</sup> In conformance with the XML specification

In the above, note that elements can have empty content (<rings></rings>). The shortcut for empty content is the tag string followed by a "/" character. This means that in the above, "<rings></rings>" can be replaced with "<rings/>".

Conversly, elements that overlap and are not properly nested to form a tree structure are not permissible. The XML content will be treated as *not* "well-formed" and the XML parser will throw-up errors and parsing will terminate.

The following will simply not work, because the start and end tags overlap and are not properly nested:

---

```
<planet> <-----+
      Earth      |
      <satellite> <---X---+
            Moon  | | OVERLAP NOT ALLOWED
</planet> <-----+ |
      </satellite> <-----+
```

---

**A quick tip:** The Microsoft Internet Explorer can check an XML document for "well-formedness". Just right-click the XML file in the explorer and open it with IE. If any errors are caught, IE will show its location along with the error description.

**Another tip:** You should use tabs (or spaces) to visually represent the nesting as shown in the example above. This will greatly help while manually editing the document. The XML parser strips any starting and ending tabs and spaces between elements and tags while those located within the content are passed on.

## 2. Attributes

*Attributes* can (but not always) form part of elements. Attributes are "tags" and corresponding values located *within* the start XML tag that are usually used to specify additional information about an element. Here are some examples of attributes:

---

```
<satellite type="artificial">SkyLab</satellite>
"type" IS AN ATTRIBUTE WITH VALUE "artificial"

<train direction="up" distance="142" unit="Km">
  Deccan Queen
</train>
ATTRIBUTE "direction" HAS THE VALUE "up" AND SO ON...
```

---

Attributes are always located within start tags (yes, end tags cannot contain attributes) and any number of them can appear within a start tag in *any* order.

Attribute values are always quoted (""") irrespective of their type (strings, integers, real, Boolean etc.), not to mention that they are case-sensitive too. Note that any starting and ending spaces and tabs within the attribute value (between quotes) may have some significance. If attributes do not conform to the above criteria, the XML document will not be treated as "well-formed."

### 3. Valid documents

The above rules concerning Elements and Attributes define the XML *syntax* but do not put any other restriction on what is and what is not allowed in a particular XML document *instance*.

For example, in the case of swCP3, the string “CPTag” is used to represent Custom Property Names. Now, if somebody editing the swCP3.xml document unknowingly (or knowingly) uses the string “CPName” to represent Custom Property Names, the XML parser will still treat the document as “well-formed”, and this mistake will not be caught by it (the parser). This defeats one of the main reasons for using XML since the person writing the application will have to do the (error) checking.

Enter the concept of a “Valid” document. Here, the XML document is “validated” against a set of rules (specified by the application designer) that impose restrictions on some (or all) aspects of an XML document instance. These may include things like what “tags” are allowed, what values certain tags and attributes can take, if any tags (and elements) are compulsory or optional, or how many times a particular element is allowed.

There are several methods of validating an XML document. In the case of swCP3, an XML “Schema” is used. A Schema is an (yet another) XML document that contains standard definitions and rules applicable to a particular XML document.

This means the XML parser checks a particular XML document instance for conformance with the XML specification for “well-formedness” and optionally – if required by the application – against a Schema for “validity.”

### 4. Namespaces

Describing content in this way leads to a particular problem. Take the case of the string “CPTag” used to represent a certain content within the context of swCP3. There is no way of knowing if the same tag (string) is used to represent something else in some other document. This may lead to confusion if someone lands-up with two XML documents having common tag strings that mean different things. The problem is compounded with common tags like <name></name> or <item></item>. No particular “owner” can be assigned to nor restrictions imposed on the usage of such tag names.

This is where the concept of “namespaces” helps out. Namespaces determine who “owns” a particular instance of a particular tag. From the user perspective, all that changes is that a namespace string has to be attached in front of the tag string with a “:” (colon) character between them.

---

```
<swCP3:CPIItem>A Test</swCP3:CPIItem>
```

namespace is “swCP3” and owns this particular instance of “CPIItem”

```
<IR:train glow:type=“Steam”>
  That old train
</IR:train>
```

tag “train” is located in the namespace “IR” while the attribute “type” is located in the namespace “glow”

---



Yes, they are required for both start and end tags, and attribute names can have them too, and as with everything else, namespaces are case-sensitive.

## **5. Benefits of using XML**

XML is a simple and structured means of describing content that enables easy processing and searching of textual content with computers (though not necessarily by humans). It also eases automated data interchange between diverse systems. It simplifies error checking by validating an XML document against a Schema (or through some other mechanism) that specifies rules governing what is actually allowed. It also permits straightforward extension of application functionality by allowing addition of new tags to express the new functionality with minimum disruptions to the overall existing application structure. The above are the main reasons for choosing XML for swCP3 – not to mention all the hype concerning it!