The Progress Electronic Magazine

In this issue:

Did you sign up to receive this E-Zine? Send email to scott_auge@yahoo.com to subscribe! It's free!

Though intended for users of the tools provided by Progress Software Corporation, this document is NOT a product of Progress Software Corporation.

Publisher's Statement:

A great big "Thanks!" to all the people who mentioned grammar errors and offered proof-reading help. I am not someone who turns down help when it is offered, so if you have ideas and such to make this a better E-Zine, do not hesitate to offer them. It is nice to see that there are people out there who desire to help me while I offer this magazine to help others.

I would also like to announce the formation of my new company called Amduus Information Works. It is important to mention this because Amduus Information Works will be assisting in the distribution of this E-Zine by making money for Scott so he can continue to do this! Hopefully Amduus Information Works will help me achieve the next step of my career, as well as provide products and services to people out there who wish to grow their companies and increase their successes.

How did I come up with such a name for my new company? Read the "A simple naming program" article in this issue. Maybe you will want to use it to name your child? =)

Also in this issue is the use of parameter records to configure your application. As a contractor I have encountered many Progress applications, and in my opinion a lot of them have TOO MUCH HARD CODING in them. So here is my contribution to better programming!

To your success,

Scott Auge
Founder, Amduus Information Works

Main Article: Configuration Options for your Application

Written by Scott Auge scott_auge@yahoo.com

Quite often as we code, we will need to set various values representing perhaps a dollar amount, or a high water mark, or a default directory within our applications. In smaller applications with little customization and integration, one might be inclined to just hard code such values.

This would be unfortunate, because a lot of times the program grows as the company grows. That is, it becomes more complex as the company defines and re-defines it's business processes, becomes integrated with other applications, and is worked on by numerous programmers working on the application – each knowing that little detail of what was hard coded.

Some examples of how I (and my co-workers) have used configuration parameters:

- Default log files
- MQ Series Queue Names
- Host Name of a machine
- URLs to other web applications
- High Water Marks for counts or other types of integers
- Decision point values for dollar figures

Though this may look a lot like the WebState table described in Issue 2, the purpose is distinctly different. The WebState table was meant to store data for a given session in the application as used by that particular user. This table is meant to parameterize the application for ALL users. The WebState table could be thought of as operational data within a transaction or activity of the user. This table is meant to store data to be used as configuration information for the application.

Below is a report for the Parms table. An explanation of it's fields follows.

```
05/05/01 11:12:51       PROGRESS Report
Database: /db/srvexp/data/srvexp (PROGRESS)




================================================================================
============================== Table: Parms ==================================

              Table Flags: "f" = frozen, "s" = a SQL table


Table                          Dump     Table Field Index Table
Name                           Name     Flags Count Count Label
------------------------------ -------- ----- ----- ----- --------------------
Parms                          parms              4     1 Parameters

  Description: Parameters
 Storage Area: N/A


============================== FIELD SUMMARY ==================================
============================== Table: Parms ==================================

Flags: <c>ase sensitive, <i>ndex component, <m>andatory, <v>iew component

Order Field Name                Data Type   Flags Format          Initial
----- ------------------------- ----------- ----- --------------- ----------
   20 Application                char        i     x(8)
   30 GroupName                  char        i     x(35)
   40 ParmName                   char        i     X(35)
   50 ParmValue                  char              x(60)

Field Name                     Label                  Column Label
------------------------------ ---------------------- ----------------------
Application                    Application            Application
GroupName                      Group Name             Group Name
ParmName                       Parm Name              Parm Name
ParmValue                      Parm Value             Parm Value


============================== INDEX SUMMARY ==================================
============================== Table: Parms ==================================
```

```
Flags: <p>rimary, <u>nique, <w>ord, <a>bbreviated, <i>nactive, + asc, - desc

Flags Index Name                       Cnt Field Name
----- -------------------------------- --- --------------------------------
pu    ParmsKey                           3 + Application
                                           + GroupName
                                           + ParmName

** Index Name: ParmsKey
 Storage Area: N/A


============================== FIELD DETAILS ==============================
============================== Table: Parms ===============================

** Field Name: Application
  Description: Application Name
         Help: Enter Application Name

** Field Name: GroupName
  Description: Used For Grouping Various Parameters Together
         Help: Enter Group Name

** Field Name: ParmName
  Description: Parameter Name
         Help: Enter Parameter Name

** Field Name: ParmValue
  Description: Parameter Value; Used like a Name/Value Pair
         Help: Enter Parameter Value
```

The two simplest fields to understand are the ParmName and ParmValue fields. One can think of them as a variable name and the value that is stored within it. Both are character fields, so if you are storing logical, decimal, or integer values, they will need to be converted to the proper type.

The next field most appropriate to speak about would be the Application field. A lot of times, a system is composed of multiple processes. Some are behind the scenes, others are applications that manipulate the database through one type of interface or another, such as GUI or the Web.

Sometimes, we want to set up parameters that effect one application and not another, or parameters that are meant for a certain background job. One can name these parameters by the application – so for example, if there is a background process that has a log file, the name of the parameter might be “LogFile”. If another background process is created that also needs a log file,

one would either need another name for the parameter, or categorize the parameter by application name.

Within an application, there may be a set of parameters associated with the same subsystem or operation, for example a credit card interface. Within an application "WebCatalog" there might be a GroupName of "CreditCard" with various parameter names such as "DialInNumber", "TerminalID", "LogFile", etc. This might differentiate these parameters from another set of parameters associated with user limits.

Since Progress supports multiple types of user interfaces, an editor for these parameter types is not provided. But following are the programs that do the bare bones manipulation of parameter records.

The program that writes parameters:

```
/*
 * WriteParms.p
 * Written by Scott Auge
 *
 * Set a parameter
 *
 */

DEF VAR RCSVersion AS CHARACTER INIT "$Header:
/home/appl/marketplaza/src/Parms/
RCS/WriteParms.p,v 1.1 2001/05/11 06:47:50 root Exp $" NO-UNDO.

DEF INPUT PARAMETER pApplication   AS CHARACTER NO-UNDO.
DEF INPUT PARAMETER pGroupName    AS CHARACTER NO-UNDO.
DEF INPUT PARAMETER pParmName        AS CHARACTER NO-UNDO.
DEF INPUT PARAMETER pParmValue       AS CHARACTER NO-UNDO.
DEF OUTPUT PARAMETER pError      AS CHARACTER NO-UNDO.

{Error.inc}

/* Determine if there is an existing parameter.  If so, we will just over-
 * write the values.  If there is not, we will create a record.  Note we look
 * up the record with a NO-WAIT option in case it is used by some other
 * program.  It is up to the calling program to determine what it should do
 * under this condition.  (Recommended is to pause and try again.)
 */

FIND Parms EXCLUSIVE-LOCK
WHERE Parms.Application = pApplication
  AND Parms.GroupName = pGroupName
  AND Parms.ParmName = pParmName
  NO-WAIT NO-ERROR.

IF NOT AVAILABLE Parms THEN
  IF LOCKED Parms THEN DO:
    ASSIGN
    pError = {&LOCKED}.
    RETURN.
```

```
  END. /* NOT AVAILABLE AND LOCKED! */
  ELSE DO:
    CREATE Parms.
    ASSIGN
    Parms.Application = pApplication
    Parms.GroupName = pGroupName
    Parms.ParmName = pParmName.
  END.

ASSIGN
Parms.ParmValue = pParmValue.

ASSIGN
pError = {&NOERROR}.
```

The program that will read a parameter:

```
/*
 * ReadParm.p
 * Written by Scott Auge
 *
 * Read a parameter
 *
 */

DEF VAR RCSVersion AS CHARACTER INIT "$Header:
home/appl/marketplaza/src/Parms/RCS/ReadParms.p,v 1.1 2001/05/11
06:47:39 root Exp $" NO-UNDO.

DEF INPUT PARAMETER pApplication   AS CHARACTER NO-UNDO.
DEF INPUT PARAMETER pGroupName     AS CHARACTER NO-UNDO.
DEF INPUT PARAMETER pParmName      AS CHARACTER NO-UNDO.
DEF OUTPUT PARAMETER pParmValue    AS CHARACTER NO-UNDO.
DEF OUTPUT PARAMETER pError        AS CHARACTER NO-UNDO.

{Error.inc}

FIND Parms NO-LOCK
WHERE Parms.Application = pApplication
  AND Parms.GroupName = pGroupName
  AND Parms.ParmName = pParmName
  NO-WAIT NO-ERROR .

IF NOT AVAILABLE Parms THEN
  IF LOCKED Parms THEN DO:
    ASSIGN pError = {&LOCKED}.
    RETURN.
  END.
  ELSE DO:
    ASSIGN pError = {&NORECORD}.
    RETURN.
  END.

ASSIGN pParmValue = Parms.ParmValue.
```

```
ASSIGN pError = {&NOERROR}.
```

A quick test program to call the program that writes parameters:

```
DEF VAR pError AS CHAR.

RUN Parms/WriteParms.p (
INPUT "TestApplication",
INPUT "TestGroup",
INPUT "TestName",
INPUT "TestValue",
OUTPUT pError
).
```

A quick test program that will read a parameter:

```
DEF VAR pError AS CHAR.
DEF VAR pValue AS CHAR.

RUN Parms/ReadParms.p (
INPUT "TestApplication",
INPUT "TestGroup",
INPUT "TestName",
OUTPUT pValue,
OUTPUT pError
).

disp pError pValue.
```

Of course, to simplify calls to these functions, one could create include files that will define default Application and GroupName parameters for the procedure. Hence the programmer only needs to remember the variable that is to be populated with the parameter, and the parameter name.

About the author: Scott Auge is the founder of Amduus Information Works. He has been programming in the Progress environment since 1994. His work have included E-Business initiatives and focuses on web applications on UNIX platforms.
scott_auge@yahoo.com

Article: A simple naming program

Written by Scott Auge scott_auge@yahoo.com

As I searched for a new name for my company, I soon discovered why all the new companies out here in Silicon Valley have taken to funky names… all the others are used already! I was going to name it NatComp Research, but it appears someone in Germany has already taken NatComp.com. So goes the story for other names I came up with.

So a name generator is in order. This is a simple program that defines a set of syllables from the English language (though not all of them) and jumbles them together to create words. It makes a lot of "noise" but it also comes up with some interesting combinations that really "sing" too.

One can add additional syllables by listing them in the Syllables variable. There are multiple ASSIGN statements used to help with readability and to keep the syllables a bit in alphabetical order. This should ease maintenance concerns.

The number of words generated can be adjusted by the value in the "DO j" loop.

The number of syllables within the words generated can by adjusted by the value in the "DO i" loop.

Notice that the output file is oriented to the UNIX file system. Be sure to change it to a Windows file naming convention if running the program on Windows.

```
DEF VAR Syllables AS CHARACTER NO-UNDO.
DEF VAR i AS INTEGER NO-UNDO.
DEF VAR j AS INTEGER NO-UNDO.

DEF VAR w AS CHARACTER NO-UNDO.

ASSIGN Syllables =
"a,ab,aw,ae,at,au,ap,as,ad,af,ag,ah,aj,ak,al,az,ax,ac,av,ab,an,am".

ASSIGN Syllables = Syllables + "," +
"b,ba,be,bo,br,bu,by,bi".

ASSIGN Syllables = Syllables + "," +
"c,ca,ce,ch,ci,ck,cl,co,cr,cu,cy".

ASSIGN Syllables = Syllables + "," +
```

```
"d,da,de,di,do,dr,du,dy".

ASSIGN Syllables = Syllables + "," +
"e,ea,eb,ec,ed,ef,ek,el,em,en,ep,eq,er,es,et,ev,ew,ex,ez".

ASSIGN Syllables = Syllables + "," +
"f,fa,fi,fe,fo,fr,fu,fy".

ASSIGN Syllables = Syllables + "," +
"g,ga,ge,gh,gi,go,gr,gu,gy".

ASSIGN Syllables = Syllables + "," +
"h,ha,he,hi,ho,hu,hy".

ASSIGN Syllables = Syllables + "," +
"i,ia,ib,ic,id,if,ig,ik,il,im,in,ip,iq,ir,is,it,iv,ix,iz".

ASSIGN Syllables = Syllables + "," +
"j,ja,je,jo,jr,ju,jy".

ASSIGN Syllables = Syllables + "," +
"k,ka,ke,kh,ki,kn,ko,kr,ku,ky".

ASSIGN Syllables = Syllables + "," +
"l,la,le,li,lo,lu,ly".

ASSIGN Syllables = Syllables + "," +
"m,ma,me,mi,mn,mo,mr,mu,my".

ASSIGN Syllables = Syllables + "," +
"n,na,ne,ni,no,nu,ny".

ASSIGN Syllables = Syllables + "," +
"o,oa,ob,oc,od,of,og,oh,oj,ok,ol,om,on,oo,op,oq,or,os,ot,ov,ow,ox,oz".

ASSIGN Syllables = Syllables + "," +
"p,pa,pe,pi,pl,po,pp,pr,pu,py".

ASSIGN Syllables = Syllables + "," +
"q,qu".

ASSIGN Syllables = Syllables + "," +
"r,ra,re,ri,ro,ru,ry".
```

```
ASSIGN Syllables = Syllables + "," +
"s,sa,se,sh,si,so,sp,st,su,sw,sy".

ASSIGN Syllables = Syllables + "," +
"t,ta,te,th,ti,to,tr,tu,ty".

ASSIGN Syllables = Syllables + "," +
"u,ub,uc,ud,uf,ug,uh,uj,uk,ul,um,un,up,us,ut,uv,uz".

ASSIGN Syllables = Syllables + "," +
"v,va,ve,vi,vo,vu,vy".

ASSIGN Syllables = Syllables + "," +
"w,wa,we,wh,wi,wo,wr,wu,wy".

ASSIGN Syllables = Syllables + "," +
"x,xa,xe,xi,xo,xu,xy".

ASSIGN Syllables = Syllables + "," +
"y,ya,ye,yi,yo,yu".

ASSIGN Syllables = Syllables + "," +
"z,za,ze,zi,zo,zu,zy".

OUTPUT TO /tmp/3CompanyNames.txt.

DO j = 1 TO 50000:  /* Number of words */

  ASSIGN w = "".

  DO i = 1 TO 3:  /* Number of syllables */

    ASSIGN w = w +
               ENTRY(RANDOM(1, NUM-ENTRIES(Syllables)), Syllables).

  END. /* DO i */

  PUT w FORMAT "x(40)" "~n".

END. /* DO j */

OUTPUT CLOSE.
```

About the author: Scott Auge is the founder of Amduus Information Works. He has been programming in the Progress environment since 1994. Works have included E-Business initiatives and focuses on web applications on UNIX platforms. scott_auge@yahoo.com

Product Updates and Releases:

Do you have a new product or release of software? Let me know and it will be included in the latest E-Zine.

Re-sale Opportunity from Amduus Information Works.

Amduus Information Works announces the availability for re-sale of a software product called Market Plaza.

Market Plaza Suite is composed of the following modules:

- *Survey Presentation* – Allows the company to present surveys to the market place and to tabulate the results. Surveys can be garnered from people visiting the site, or through the use of the Press Room which can have emails sent out to customers. It could also provide leads or extract knowledge about the marketplace. This can be used internally as well as externally to the company.

- *Project Collaboration* – Often sub-contractors or people within the customer's as well the purchaser's organization need to collaborate on projects, installations, and configurations. This is a way for information to be distributed that normally would have been sequestered away in someone's email box.

- *Message Forum* – A more open form of Project Collaboration, sort of like the Yahoo Clubs. This can be used internally as well as externally to the company.

- *Press Room* – This module allows the company to organize press releases and other corporate information. It will also allow the mass mailing of information to customers based on a list of available email addresses. Useful for presenting survey requests, product information and updates, etc. Also will present such information on a web page.

- *Electronic Catalog* – A means to present products for sale to the market place, as well to receive orders from the market place through the web site.

- *Service Center* – Allows the customer to open an work ticket for a product performance issue or for a help call. Such contacts could be for requests of service work on a product, for disputes of billing, or requests of help. Problems and purpose of the call is configurable to be delivered to the appropriate employee. This can be used both internally and externally to the company.

- *Product Configuration* – Some products cannot just be purchased from a catalog because there is a collection of options that need to be chosen. The Product Configuration module allows the presentation of products and a means to interact with the customer via questions posed to them to which they can choose options to construct a model of the product they wish to purchase. Examples could be telephone switching equipment to measurements of jeans. This could be used internally and externally to the company.

- *Knowledge Base* – Some companies have more complicated services or products than others. Questions about the product can be based on model and design and placed into a knowledge base that customers can call upon to gain information. Such things as questions and answers, location for documentation about a particular product, and miscellaneous information can be placed in this package. This can be used internally as well as externally to the company.

  Contact Scott Auge at scott_auge@yahoo.com

Jobs Available:

Do you have a job opening available? Let me know and it will be included in the latest E-Zine. Price is $10.00 per listing per issue.

Contractors/Consulting Companies Available:

If you do work in the Progress world – let me know and I will be able to include you! Price is $10.00 per listing per issue.

Analysts Express Inc.

The Reliable Source for I.T. Consulting and Recruiting
Contact: James Arnold @ 888/889-9091

02-1

Amduus Information Works
4506 Carlyle Court Suite 712
Santa Clara, CA 95054
408-980-8447
scott_auge@yahoo.com

Creation of modules and products for re-sale. Internet/Intranet programming for E-Business in the marketing/manufacturing/service/law enforcement industries.

Jay A. Martin
DataChase, LLC
www.data-chase.com
info@data-chase.com

Providing contract programming and writing services.

01-07

Community Announcements:

A place to announce Progress User Groups, Open Source Exchanges, etc. No charge!

Wonder where to find those Progress marketing articles are?

http://www.progress.com/analyst/
http://www.progress.com/profiles/index.htm
http://www.progress.com/success/index.htm

Publishing Information:

Scott Auge publishes this document. I can be reached at scott_auge@yahoo.com.

This electronic magazine is to be published once a month.

Currently there are over 300 subscribers and companies that receive this mailing!

Amduus Information Works assists in the publication of this document:

Amduus Information Works
4506 Carlyle Court Suite 712
Santa Clara, CA 95054

Article Submission Information:

Currently, payment for an article is not possible. But – articles CAN enhance your prestige and name recognition – and these things can translate into money!

Please submit your article in Microsoft Word format or as text. Please include a little bit about yourself for the About the Author paragraph.

Looking for technical articles, *marketing Progress* articles, articles about books relevant to programming/software industry, white papers, etc.

Upcoming Articles:

July 2001
Sending and receiving email with your progress application.
How many Webspeed agents?