# producer

| COLLABORATORS | | | |
|---|---|---|---|
| | *TITLE* :<br><br>producer | | |
| *ACTION* | *NAME* | *DATE* | *SIGNATURE* |
| WRITTEN BY | | August 30, 2024 | |

| REVISION HISTORY | | | |
|---|---|---|---|
| NUMBER | DATE | DESCRIPTION | NAME |
| | | | |

# Contents

# Chapter 1

# producer

## 1.1 producer.adoc

```
 GetProducer()
 FreeProducer()
 LoadDesignerData()
 FreeDesignerData()
 OpenProducerWindow()
 CloseProducerWindow()
 SetProducerWindowFileName()
 SetProducerWindowAction()
 SetProducerWindowLineNumber()
 ProducerWindowUserAbort()
 ProducerWindowWriteMain()
 AddLocaleString()
 FreeLocaleStrings()
 WriteLocaleCT()
 WriteLocaleCD()
```

## 1.2 producer.library/GetProducer

```
NAME
    GetProducer - Add a dynamic node host.

SYNOPSIS
    pn = GetProducer ( void );
    d0

    struct ProducerNode * pn;

FUNCTION
    This function creates a new Producer.

INPUTS

RETURNS
    pn      - pn to be used in just about every Producer operation.
```

```
EXAMPLE

    pn = GetProducer();
    if (pn)
        {
        /*

        */
        FreeProducer(pn);
        }

SEE ALSO
    FreeProducer()
```

## 1.3 producer.library/FreeProducer

```
NAME
    FreeProducer - Free all data and close down producer

SYNOPSIS
     void FreeProducer ( pn );
                          A0

     struct ProducerNode *pn;

FUNCTION
    This gets rid of all data

INPUTS
    pn - ProducerNode allocated.

EXAMPLE

    pn = GetProducer();
    if (pn)
        {
        /*

        */
        FreeProducer(pn);
        }

RETURNS

SEE ALSO
    GetProducer()
```

## 1.4 producer.library/LoadDesignerData

```
NAME
    LoadDesignerData - Load a file into a ProducerNode.
```

```
SYNOPSIS
    res = LoadDesignerData ( pn, filename  );
                        a0  A1

    long res;
    struct ProducerNode *pn;
    char * filename;

FUNCTION
    Load a designerfile into the ProducerNode supplied.

INPUTS
    pn       - Your ProducerNode.
    filename - Filename to load

RETURNS

    res - 0 if OK, error code otherwise.

SEE ALSO

    FreeDesignerData()
```

## 1.5 producer.library/FreeDesignerData

```
NAME
    FreeDesignerData - frees all data loaded into a ProducerNode

SYNOPSIS
    ExpungeDataBases ( pn );
                       A0

    struct ProducerNode *pn;

FUNCTION
    Free all allocated memory for data structures.

INPUTS
    pn      - Your ProducerNode

SEE ALSO
    LoadDesignerData()
```

## 1.6 producer.library/OpenProducerWindow

```
NAME
    OpenProducerWindow - Open producers window.

SYNOPSIS
    res = OpenProducerWindow( pn, title );
                              A0  A1
```

```
    long res;
    struct ProducerNode * pn;
    char * title;

FUNCTION
    This function attempts to open a window associated with the
   supplied ProducerNode. This window is now the standard look for
   the Producers.

RETURNS
    res - 1 if success, 0 if failure.

SEE ALSO
    CloseProducerWindow()
```

## 1.7 producer.library/CloseProducerWindow

```
NAME
    CloseProducerWindow - Close Window associated with a producer.

SYNOPSIS
    CloseProducerWindow (pn);
                         A0

    struct ProducerNode *pn;

FUNCTION
    This function is used to close your producer window.
    It will be done automatically when you freen your
    producer.

INPUTS
    pn ProducerNode handle.

RETURNS

SEE ALSO
    OpenProducerWindow()
```

## 1.8 producer.library/SetProducerWindowFileName

```
NAME
    SetProducerWindowFileName - Set text in gadget in producer window..

SYNOPSIS
    SetProducerWindowFileName( pn, filename );
                               A0  A1

    struct ProducerNode *pn
    char *filename

FUNCTION
```

```
    This function is used to change the contents of the FileName
    gadget in the producer window.

INPUTS
    pn - Producer.
    filename - Text to place.

RETURNS

SEE ALSO
    SetProducerWindowAction(), SetProducerWindowLineNumber()
```

## 1.9 producer.library/SetProducerWindowAction

```
NAME
    SetProducerWindowAction - Set text in gadget in producer window..

SYNOPSIS
    SetProducerWindowAction( pn, action );
                             A0  A1

    struct ProducerNode *pn;
    char *action;

FUNCTION
    This function is used to change the contents of the Action
    gadget in the producer window.

INPUTS
    pn - Producer.
    action - Text to place.

RETURNS

SEE ALSO
    SetProducerWindowFileName(), SetProducerWindowLineNumber()
```

## 1.10 producer.library/SetProducerWindowLineNumber

```
NAME
    SetProducerWindowLineNumber - Set text in gadget in producer window..

SYNOPSIS
    SetProducerWindowLineNumber( pn, number );
                                 A0  D0

    struct ProducerNode *pn;
    long number;

FUNCTION
    This function is used to change the contents of the LineNumber
    gadget in the producer window.
```

```
INPUTS
    pn - Producer.
    LineNumber - Number to place.

RETURNS

SEE ALSO
    SetProducerWindowAction(), SetProducerWindowLineNumber()
```

## 1.11 producer.library/ProducerWindowUserAbort

```
NAME
    ProducerWindowUserAbort - See if user has aborted the production.

SYNOPSIS
    ret = ProducerWindowUserAbort( pn );
                                   A0

    struct ProducerNode *pn;
    long ret;

FUNCTION
    This function is used to change test whether the user has requested
    an abort by the closegadget, the abortgadget or by pressing 'A'.

INPUTS
    pn - Producer.

RETURNS
    1 if user aborts, 0 otherwise.
```

## 1.12 producer.library/ProducerWindowWriteMain

```
NAME
    ProducerWindowWriteMain - See if user wishes to overwrite main file.

SYNOPSIS
    ret = ProducerWindowWriteMain( pn, filename );
                                   A0  A1

    struct ProducerNode *pn;
    char * filename;
    long ret;

FUNCTION
    This function is used to find out if the already exists a Main
    file. If it does then this will allow the user to choose whether
    to overwrite it or not.
    None and All are handled as well.

INPUTS
```

```
    pn - Producer.
    filename - Prospective Main filename.

RETURNS
    1 if user selects write file, 0 otherwise.
```

## 1.13 producer.library/AddLocaleString

```
NAME
    AddLocaleString - Add string to locale list.

SYNOPSIS
    ret = AddLocaleString( pn, string, label, comment );
                           A0  A1      D0     D1

    struct ProducerNode *pn;
    long ret;

FUNCTION
    This function is used to add a string to the list of strings that
    will be written in any .cd or .ct file you produce in the
    Producer.
    This list is available in the ProducerNode and should be converted
    into the default strings in your program. This list also contains
    extra strings defined in the locale window in the Designer.
    The first item in the list has label with value 0.

INPUTS
    pn - Producer.
    string - Text.
    label - Label to be used in code.
    comment - Comment to be placed in .cd and .ct files.

RETURNS
    1 if OK, 0 otherwise.
```

## 1.14 producer.library/FreeLocaleStrings

```
NAME
    FreeLocaleStrings - Free locale list.

SYNOPSIS
    FreeLocaleStrings( pn );
                       A0

    struct ProducerNode *pn;

FUNCTION
    This will free all locale strings, you will probably never
    need it.

INPUTS
```

```
    pn - Producer.

RETURNS
```

## 1.15 producer.library/WriteLocaleCT

```
NAME
    WriteLocaleCT - Write .ct file.

SYNOPSIS
    WriteLocaleCT( pn );
                   A0

    struct ProducerNode *pn;

FUNCTION
    This will write a .ct file for use in flexcat etc.

INPUTS
    pn - Producer.

RETURNS
    1 if OK, 0 otherwise.
```

## 1.16 producer.library/WriteLocaleCD

```
NAME
    WriteLocaleCD - Write .CD file

SYNOPSIS
    WriteLocaleCD( pn );
                   A0

    struct ProducerNode *pn;

FUNCTION
    This will write a .cd file for use in flexcat etc.

INPUTS
    pn - Producer.

RETURNS
    1 if OK, 0 otherwise.
```