Supportpac CA8I User Guide

CICS Transaction Server for OS/390 and z/OS - A utility to convert DOCUMENT templates to data-only programs for improved performance

Version 1.1

13 December 2002

Mark Cocker (mark_cocker@uk.ibm.com)

and

Peter Havercan (havercan@hursley.ibm.com)

CICS Development
IBM UK Labs Ltd

Take Note!

Before using this report be sure to read the general information under "Notices".

First Edition, December 2002

This edition applies to Version 1.1 of “A utility to convert DOCUMENT templates to load modules for improved performance” and to all subsequent releases and modifications unless otherwise indicated in new editions.

Table of Contents

Preface

This Supportpac provides a utility to convert an input file (text or binary) into an MVS Assembler program that, once compiled and bound into a data-only program, can be used as a document template with the EXEC CICS DOCUMENT APIs. When compared to sequential files or PDS members, templates that are based on load modules provide;

- **a significant reduction in disk I/O and latency** because data-only program templates are loaded once and cached in memory automatically by CICS just like other programs.
- **reduced CPU** usage when inserting a template into a document.
- **an improved method of systems management** because the template can be managed through well-understood system programing interfaces for programs. For example, the template can be set in service, out of service, and a new version of a template installed by issuing the command:
  CEMT SET PROGRAM(<template>) NEWCOPY
- **reduced document size** by removing leading spaces, trailing spaces, and carriage return and line-feed characters where appropriate. This is especially useful for XML templates, where spaces are added for ease of development and debugging but are an unnecessary overhead in document size, transmission, and parser processing.

The anticipated users of this Supportpac are application developers and system programmers responsible for deploying new DOCUMENT templates.

Supportpac Requirements

- REXX support to be installed in the MVS/ESA batch environment.
- CICS Transaction Server for OS/390 V1R3 or above to run the Supportpac samples.

Summary of Amendments

Date	Version	Changes
13 December 2002	1.1	Initial release

Bibliography

CICS Application Programming Reference, Release 3, SC33-1688-38.

CICS Application Programming Guide, Release 3, SC33-1687-37, Part 3 Application design, Chapter 15 Using CICS documents.

Chapter 1. Introduction

CICS TS V1R3 delivered a new EXEC CICS DOCUMENT application programming interface to provide a set of flexible services for creating and managing generic documents. The documents are generic because CICS does not dictate or interpret their contents (except for codepage conversion of text where requested). Typically the DOCUMENT API is used to create HTML or XML responses to be returned to a client application, such as a browser, a servlet or web services client, but it could be put to many other uses.

The functions provided by the DOCUMENT API are fully described in the CICS Application Programming Guide and CICS Application Programming Reference. In summary, its flexibility comes from the ability to;

- **construct a document in multiple stages** – for example a program can create a new document, insert an XML header, then execute an SQL query and for each row in the result set insert XML elements, and finally end the document by inserting XML end of document elements.

- **construct a document out of sequence** by inserting named bookmarks into the document and subsequently inserting content at the appropriate bookmark.

- **insert templates** enabling a separation of dynamic content from static data. Templates are defined as CICS DOCTEMPLATE resources in the CSD and can be stored on various medium – see the table below for a comparison.

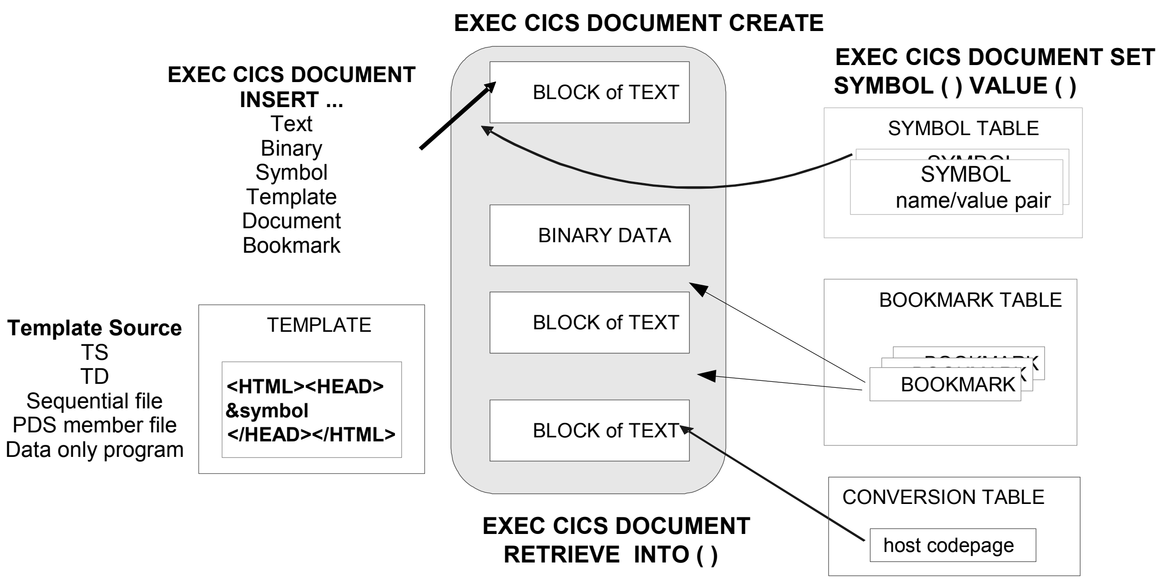
- **replace symbols in templates with values specified at runtime** allowing for the merging of static and dynamic content.

- **create documents greater than 32K** in length - clients connecting via CICS Web services, WebSphere MQ, or IIOP have the ability to return greater than 32K responses.

- **automatic codepage conversion** from the template codepage to that required by the client (e.g. ASCII).

DOCTEMPLATE resource	***Storage***	***Maximum size***	***Initialised, and refreshed by***	***Performance***	***Notes***
TS queue	Memory	Varies according to storage definition. If template larger than a storage record (typically 32763 bytes) template needs to be written as multiple queue items.	User written program(s). See CICS Supportac CA8G for examples.	Excellent	TS queue access serialized.
	Coupling Facility			Good	
	Auxiliary			Acceptable	
TD queue	Memory			Excellent	
	Coupling Facility			Good	
	File			Acceptable	
Exit program	Flexible	Practically none		Depends on storage	
CICS file	VSAM	Practically none	CICS	Good	
PDS member	File	Practically none	CICS, reinstall DOCTEMPLATE for refresh	Moderate	
Data-only program	File then cached	Practically none	CICS	Excellent	PROGRAM SPI

The diagram below illustrates that the DOCUMENT API provides verbs CREATE, INSERT, SET and RETRIEVE to manage a document. The lifecycle of the document is automatically managed by CICS - once a document is created it remains available until the task that created it ends, at which time it is deleted.



Document Templates

Document templates are a convenient mechanism to separate the static content of a document from the dynamic data which is generated as part of business logic code. Document symbol substitution allows symbols contained in the document template to be replaced with dynamic data. This is similar in concept to separating presentation logic from business logic – where BMS maps describe the presentation of a 3270 screen and are separate from the application code that is responsible for dynamic data.

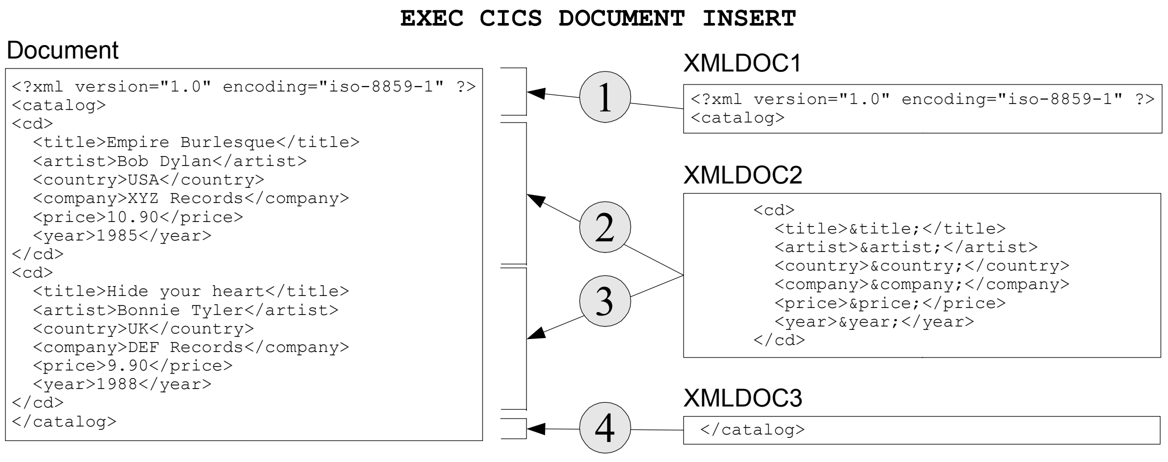
In the example below a document is created and three templates are inserted in the following order;

1. XMLDOC1 as-is,

2. XMLDOC2 replacing the symbols (title, artist, country, company, price, and year) with dynamic values,

3. XMLDOC2 for a second time, replacing the same symbols with different values, and

4. XMLDOC3 as-is to completed the document. For more details see the XMLDOCP COBOL sample included in this Supportpac.



```
<?xml version="1.0" encoding="iso-8859-1" ?>
<catalog>
<cd>
  <title>Empire Burlesque</title>
  <artist>Bob Dylan</artist>
  <country>USA</country>
  <company>XYZ Records</company>
  <price>10.90</price>
  <year>1985</year>
</cd>
<cd>
  <title>Hide your heart</title>
  <artist>Bonnie Tyler</artist>
  <country>UK</country>
  <company>DEF Records</company>
  <price>9.90</price>
  <year>1988</year>
</cd>
</catalog>
```

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<catalog>
```

```
      <cd>
        <title>&title;</title>
        <artist>&artist;</artist>
        <country>&country;</country>
        <company>&company;</company>
        <price>&price;</price>
        <year>&year;</year>
      </cd>
```

```
 </catalog>
```

Generating data-only program templates

Document templates based on data-only programs are more efficient and easier to manage than those based on a sequential file or partitioned dataset (PDS). However the people responsible for providing the template content will generally prefer to maintain the templates in an easy to edit text format such a sequential file or PDS.

What is needed, therefore, is an easy to use process that automates;

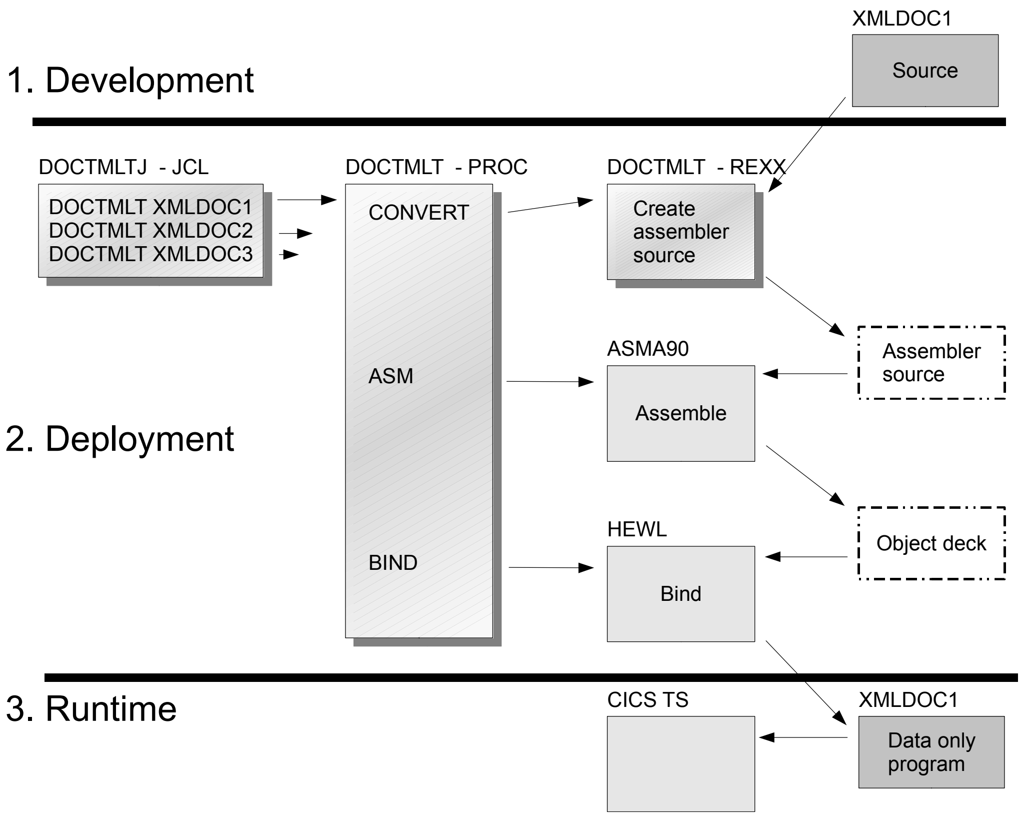
1. The conversion of templates from a sequential file or PDS to an assembler source.

2. The compiling of these programs to object decks.

3. The binding of the object decks into the final template data-only program.

This process is analogous to maintaining BMS (Basic Mapping Support) maps and compiling them into BMS mapset load modules for efficient runtime use.

The diagram below shows the suggested components and process that meets these requirements - this Supportpac supplies the DOCTMLTJ JCL, DOCTMLT PROC, and the DOCTMLT REXX EXEC.



1. Development - the source document template is maintained using any editor - typically ISPF or a workstation based editor and uploaded into a sequential file or PDS member.

2. Deployment - the DOCTMLTJ JCL is submitted to run the conversion process.

   The DOCTMLTJ JCL will start the DOCTMLT PROC for each template, passing optional parameters that may be required to tailor the conversion.

   The DOCTMLT PROC contains 3 steps
   - CONVERT runs the DOCTMLT REXX EXEC to read the template and create the assembler source.
   - ASM runs the standard MVS ASMA90 assembler to produce the object deck.

- BIND runs the standard MVS HEWL binder to bind the object deck into a data-only program.

3. Runtime - the data-only program should be placed into a dataset in the CICS DFHRPL concatenation.

   - If CICS autoinstall is enabled, the program will be picked up automatically, otherwise a static PROGRAM resource definition will need to be created and installed.

   - Create and install a DOCTEMPLATE definition.

Chapter 2. Installation

The following instructions guide you through downloading and installing the code provided with this supportpac, tailoring it for your environment, and converting the sample templates.

1. Download the Supportpac file `CA8I.ZIP` file from the IBM CICS Supportpac link on the web site:

   `http://www.ibm.com/software/ts/cics/downloads/`

2. UNZIP the `CA8I.ZIP` file into a convenient directory. The files extracted should be as follows:

`CA8I.PDF`	Documentation
`DOCTMLTJ.JCL`	JCL
`DOCTMLT.REXX`	REXX
`DOCTMLT.PROC`	PROC
`XMLDOCP`	Sample COBOL program that uses the sample templates
`XMLDOC1`	Sample XML template
`XMLDOC2`	Sample XML template
`XMLDOC3`	Sample XML template

3. Upload the DOCTMLTJ.JCL file in TEXT format to an MVS PDS used for your development JCL as member "DOCTMLTJ". Edit this file and change the 'MBR=' parameters to be the template member names, and specifying 'CPARM=' to contain conversion parameters.

   ```
   //DOCTMLTJ JOB MSGCLASS=A,NOTIFY=&SYSUID,REGION=0M
   //  EXEC DOCTMLT,MBR=XMLDOC1
   //  EXEC DOCTMLT,MBR=XMLDOC2,CPARM='COMMENTS RLS RTS'
   //  EXEC DOCTMLT,MBR=XMLDOC3
   ```

4. Upload the DOCTMLT.REXX file in TEXT format to an MVS PDS used for your development REXX EXECs as member DOCTMLT. The parameters for DOCTMLT are described in Chapter 3.

5. Upload the DOCTMLT.PROC file in TEXT format to an MVS PDS used for your development PROCs as member "DOCTMLT". Edit this file and change;

ASMBLR	to be the default assembler to use.
BINDER	to be the default binder to use.
LIB	to be the default PDS where your template files are stored.
LOADLIB	to be the default program load library for your programs used by CICS.
CPARM	to be the default parameters for the template conversion.
REXXLIB	to be the DSN where the REXX EXEC was uploaded.

```
//DOCTMLT PROC ASMBLR=ASMA90,
//       BINDER=HEWL,
//       MBR=SOURCE,
//       LIB=CICSU.DFHHTML,
//       LOADLIB=CICSU.LOAD,
//       REXXLIB=ETP.REXX,
//       CPARM='COMMENTS CRLF RSN RTS'
//*****************************************************************
//* CPARMS: CRLF = ADD CARRIAGE RETURN AND LINE FEED CHARACTERS
//*         COMMENTS = ADD COMMENTS TO THE GENERATED ASSEMBLER
//*         RLS = REMOVE LEADING SPACES FROM  THE TEMPLATE
//*         RTS = REMOVE TRAILING SPACES FROM TEMPLATE
//*         RSN = REMOVE SEQUENCE NUMBERS FROM TEMPLATE COLS 73-80
//*****************************************************************
//CONVERT  EXEC PGM=IRXJCL,
//         PARM='DOCTMLT INFILE OUTFILE &MBR ( &CPARM'
//INFILE   DD DSN=&LIB(&MBR),DISP=SHR
//OUTFILE  DD DSN=&&SOURCE,DISP=(,PASS),
//            SPACE=(CYL,(1,1)),
//            DCB=(RECFM=FB,LRECL=80,BLKSIZE=6160)
//SYSEXEC  DD DSN=&REXXLIB,DISP=SHR
```

```
//SYSTSIN  DD DUMMY
//SYSTSPRT DD SYSOUT=*
//****************************************************************
//ASM      EXEC PGM=&ASMBLR,
//         PARM=(OBJECT,NODECK,FLAG(NOALIGN)),
//         COND=(8,LE,CONVERT)
//SYSPRINT DD SYSOUT=*
//SYSPUNCH DD SYSOUT=*
//SYSLIB   DD DSNAME=SYS1.MACLIB,DISP=SHR
//SYSUT1   DD DSNAME=&&SYSUT1,UNIT=SYSDA,
//         SPACE=(CYL,(10,1))
//SYSLIN   DD DSN=&&OBJECT,UNIT=SYSDA,
//         SPACE=(CYL,(10,2)),DCB=BLKSIZE=3120,DISP=(,PASS)
//SYSIN    DD DSN=&&SOURCE,DISP=(OLD,DELETE)
//****************************************************************
//BIND     EXEC PGM=&BINDER,
//         PARM='XREF,LIST,LET',COND=(8,LE,ASM)
//SYSPRINT DD SYSOUT=*
//SYSLIN   DD DSN=&&OBJECT,DISP=(OLD,DELETE)
//SYSUT1   DD DSN=&&SYSUT1,UNIT=SYSDA,
//         SPACE=(CYL,(10,1))
//SYSLMOD  DD DSN=&LOADLIB(&MBR),DISP=SHR
```

6. Upload the XMLDOC1, XMLDOC2, and XMLDOC3 templates in TEXT format to your default template PDS.

7. Upload the XMLDOCP COBOL sample in TEXT format to your COBOL source PDS.

8. Submit the DOCTMLTJ JCL to convert the sample templates.

9. Check the DOCTMLTJ JCL output in SDSF to ensure all steps completed with a RC of 0.

10. Check the new data-only programs were created in the default program load library used by CICS.

11. Log onto the CICS region and with the authority to create and install resource definitions.

12. Create a DOCTEMPLATE resource definition for each template data-only program created above. The DOCTEMPLATE name is used to identify this resource, the TEMPLATENAME is name used by the CICS application to refer to this template, PROGRAM is the data-only program just created, and TYPE is EBCDIC or BINARY as appropriate for the template. An example is below:

```
CEDA  View DOctemplate( XMLDOC1  )
  DOctemplate     : XMLDOC1
  Group           : ETP
  DEscription     :
 FULL TEMPLATE NAME
  TEmplatename    : XMLDOC1
 ASSOCIATED CICS RESOURCE
  File            :
  TSqueue         :
  TDqueue         :
  Program         : XMLDOC1
  Exitpgm         :
 PARTITIONED DATA SET
  DDname          :
  Membername      :
 TEMPLATE PROPERTIES
  Appendcrlf      : Yes                Yes | No
  TYpe            : Ebcdic             Binary | Ebcdic
```

13. If the CICS region has program autoinstall enabled, skip this step. Otherwise create a PROGRAM resource definition for each data-only program created above. Specify PROGRAM as the data-only program name, and LANGUAGE as Assembler.

```
CEDA  View PROGram( XMLDOC1  )
 PROGram         : XMLDOC1
 Group           : ETP
 DEscription     : DOCUMENT TEMPLATE SAMPLE 1
 Language        : Assembler           CObol | Assembler | Le370 | C | Pli
 RELoad          : No                  No | Yes
 RESident        : No                  No | Yes
 USAge           : Normal              Normal | Transient
 USElpacopy      : No                  No | Yes
 Status          : Enabled             Enabled | Disabled
 RSl             : 00                  0-24 | Public
 CEdf            : Yes                 Yes | No
 DAtalocation    : Any                 Below | Any
 EXECKey         : User                User | Cics
 COncurrency     : Quasirent           Quasirent | Threadsafe
```

14. Install the DOCTEMPLATE and PROGRAM resource definitions.

15. Run your application that makes use of these DOCUMENT templates. For an sample see the XMLDOCP COBOL program below.

If the template source is modified, resubmit the DOCTMLTJ JCL. The new versions of the templates should be installed into CICS via:

```
CEMT SET PROGRAM(<template>) NEWCOPY
```

XMLDOCP COBOL Sample

The XMLDOCP COBOL program provided with this Supportpac is an example of how the templates XMLDOC1, XMLDOC2, and XMLDOC3 could be combined to produce an XML document. In the install procedure above you will have uploaded the XMLDOCP source into a suitable library and compiled the example templates into data-only programs. To run the sample:

1. Translate, compile, and bind XMLDOCP using your standard procedures.

2. Define and install a PROGRAM RESOURCE for XMLDOCP.

3. Log onto the CICS region and call the XMLDOCP program using, for example, the CECI command:

   ```
   CECI LINK PROGRAM(XMLDOCP)
   ```

4. The CICS CEEMSG output will contain the following text showing the resultant document:

```
XMLDOCP - SUCCESS
DOC-BUFFER-LEN:00000382
DOC-BUFFER....:<?xml version="1.0" encoding="iso-8859-1" ?>  <catalog>   <cd><title>Empire Burlesque</title><artist>Bob Dylan</artis
t><country>USA</country><company>XYZ Records</company><price>10.90</price><year>1985</year></cd> <cd><title>Hide your heart</title><
artist>Bonnie Tyler</artist><country>UK</country><company>DEF Records</company><price>9.90</price><year>1988</year></cd> </catalog>
```

Chapter 3. Template conversion options

The DOCTMLT REXX EXEC reads a template and generate an MVS assembler data-only program with DC statements containing the hex byte equivalent of the template.

SYNTAX: `DOCTMLT <INFILE> <OUTFILE> <CSECT> ( <options>`

The parameters are not case sensitive. The following parameters are required and order sensitive ;

• **<INFILE>**	is the file name of the template to be converted
• **<OUTFILE>**	is the file name of the generated S390 Assembler file. Note if this file already exists it will be overwritten without warning
• **<CSECT>**	is the name of the assembler CSECT name following MVS assembler CSECT naming rules, including 1-8 characters.

The <options> parameters to be specified after the '(' are optional and can be placed in any order;

• **CRLF**	add carriage return (x'0D') + line-feed (x'25') characters to the end of each template line
• **COMMENTS**	add assembler comments
• **REMOVELEADINGSPACES** or **RLS**	remove leading spaces from each line in the template Care should be taking when using this option with HTML files which can rely on leading spaces to correctly format text in <pre> tags.
• **REMOVETRAILINGSPACES** or **RTS**	remove trailing spaces from each line in the template
• **REMOVESEQUENCENUMS** or **RSN**	remove sequnce nums in cols 73-80 from each line in the template

Example using XMLDOC2

To convert the XMLDOC2 template;

```
<cd>
  <title>&title;</title>
  <artist>&artist;</artist>
  <country>&country;</country>
  <company>&company;</company>
  <price>&price;</price>
  <year>&year;</year>
</cd>
```

removing trailing and leading spaces, and generating assembler comments, DOCTMLTJ JCL should be changed as follows and submitted:

```
//DOCTMLT  JOB MSGCLASS=A,NOTIFY=&SYSUID,REGION=0M
//  EXEC DOCTMLT,MBR=XMLDOC2,CPARM='COMMENTS RLS RTS'
```

The output for this job shows a RC=0;

```
OBNAME  STEPNAME PROCSTEP    RC
OCTMLT           CONVERT   0000
OCTMLT           ASM       0000
OCTMLT           BIND      0000
```

and further down in the output we can see the following assembler listing:

```
Stmt   Source Statement
   1 * -----------------------------------------------------------------
   2 * CICS SupportPac CA8I..: Version 1.1
   3 * Generated on..........: 12 Dec 2002 - 08:10:06
   4 * Remove leading spaces.: YES
   5 * Remove trailing spaces: YES
   6 * Remove sequence nums..: NO
   7 * Add CR+LF to each line: NO
   8 * Comments..............: YES - non-printable chars converted to '.'
   9 * # bytes in input file.: 640
  10 * # bytes in prolog.....: 30
  11 * # bytes in TEMPLATE...: 154
  12 * =================================================================
  13          PUNCH ' ENTRY TEMPLATE'
  14 XMLDOC2 CSECT
  15 XMLDOC2 RMODE ANY
  16 XMLDOC2 AMODE 31
  17          DC CL9'XMLDOC2'
  18          DC CL9'&SYSDATE'
   +         DC CL9'12/12/02'
  19          DC CL5'&SYSTIME'
   +         DC CL5'08.10'
  20 * padding to align the end of TEMPLATE to an 8-byte boundary
  21          DC X'40404040404040'
  22 TEMPLATE EQU *
  23          ENTRY TEMPLATE
  24 * -----------------------------------------------------------------
  25 *     <cd><title>&title;</title><art
  26  DC X'4C83846E4CA389A393856E50A389A393855E4C61A389A393856E4C8199A3'
  27 *     ist>&artist;</artist><country>
  28  DC X'89A2A36E508199A389A2A35E4C618199A389A2A36E4C8396A495A399A86E'
  29 *     &country;</country><company>&c
  30  DC X'508396A495A399A85E4C618396A495A399A86E4C839694978195A86E5083'
  31 *     ompany;</company><price>&price
  32  DC X'9694978195A85E4C61839694978195A86E4C97998983856E509799898385'
  33 *     ;</price><year>&year;</year></
  34  DC X'5E4C6197998983856E4CA88581996E50A88581995E4C61A88581996E4C61'
  35 *     cd>.
  36  DC X'83846E40'
  37 * -----------------------------------------------------------------
  38          END TEMPLATE
```

Note that leading and trailing spaces have been removed, comments in the assembler help to check the converted template, and no carriage returns or line-feeds are present.

--- End of Document --