

**Dynamic Server Pages Guide**  
10g Release 3 (10.1.3.3.0)

March 2007

Dynamic Server Pages Guide, 10g Release 3 (10.1.3.3.0)  
Copyright © 2007, Oracle. All rights reserved.

Contributing Authors: Sandra Christiansen, Karen Johnson

Contributors: Sam White, Eva Cordes

The Programs (which include both the software and documentation) contain proprietary information; they are provided under a license agreement containing restrictions on use and disclosure and are also protected by copyright, patent, and other intellectual and industrial property laws. Reverse engineering, disassembly, or decompilation of the Programs, except to the extent required to obtain interoperability with other independently created software or as specified by law, is prohibited.

The information contained in this document is subject to change without notice. If you find any problems in the documentation, please report them to us in writing. This document is not warranted to be error-free. Except as may be expressly permitted in your license agreement for these Programs, no part of these Programs may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose.

If the Programs are delivered to the United States Government or anyone licensing or using the Programs on behalf of the United States Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the Programs, including documentation and technical data, shall be subject to the licensing restrictions set forth in the applicable Oracle license agreement, and, to the extent applicable, the additional rights set forth in FAR 52.227-19, Commercial Computer Software--Restricted Rights (June 1987). Oracle USA, Inc., 500 Oracle Parkway, Redwood City, CA 94065.

The Programs are not intended for use in any nuclear, aviation, mass transit, medical, or other inherently dangerous applications. It shall be the licensee's responsibility to take all appropriate fail-safe, backup, redundancy and other measures to ensure the safe use of such applications if the Programs are used for such purposes, and we disclaim liability for any damages caused by such use of the Programs.

Oracle, JD Edwards, PeopleSoft, and Siebel are registered trademarks of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

The Programs may provide links to Web sites and access to content, products, and services from third parties. Oracle is not responsible for the availability of, or any content provided on, third-party Web sites. You bear all risks associated with the use of such content. If you choose to purchase any products or services from a third party, the relationship is directly between you and the third party. Oracle is not responsible for: (a) the quality of third-party products or services; or (b) fulfilling any of the terms of the agreement with the third party, including delivery of products or services and warranty obligations related to purchased products or services. Oracle is not responsible for any loss or damage of any sort that you may incur from dealing with any third party.

# Table of Contents



## Chapter 1: About This Guide

Audience .....	1-1
Conventions .....	1-1
Organization .....	1-2

## Chapter 2: Understanding Dynamic Server Pages

About Dynamic Server Pages .....	2-1
Page Types .....	2-3
IDOC File .....	2-3
HCST File .....	2-3
HCSP File .....	2-3
HCSF File .....	2-4
Creating Dynamic Server Pages .....	2-4
Syntax .....	2-5
Idoc Script Tags .....	2-5
Comparison Operators .....	2-6
Special Characters .....	2-7
Referencing Metadata .....	2-7
Idoc Script Functions .....	2-8
docLoadResourceIncludes Function .....	2-8
Parameters .....	2-9
executeService Function .....	2-9
HCSF Pages .....	2-10
Development Recommendations .....	2-10
General Tips .....	2-10
HCSF Tips .....	2-11

## Chapter 3: HCSF Pages

Load Section .....	3-1
HTML Declaration .....	3-2

docLoadResourceIncludes Function . . . . .	3-2
Meta Tag . . . . .	3-2
Variables and Includes . . . . .	3-2
Data Section. . . . .	3-3
Data Section Structure. . . . .	3-3
idcformrules Tag . . . . .	3-4
isFormFinished Attribute . . . . .	3-4
resultsets Attribute . . . . .	3-4
Metadata Tags. . . . .	3-5
content Attribute . . . . .	3-5
Nested Tags . . . . .	3-5
Referencing XML Tags . . . . .	3-6
Form Elements . . . . .	3-6
ResultSets . . . . .	3-7
Example . . . . .	3-7
Repeated Tags in a ResultSet . . . . .	3-8
Nested Tags in a ResultSet . . . . .	3-8
Editing a ResultSet . . . . .	3-9
Form Section . . . . .	3-10
Form Begin . . . . .	3-10
Form Properties. . . . .	3-11
Form Fields . . . . .	3-11
DataScript . . . . .	3-11
Form Buttons. . . . .	3-12
Form End. . . . .	3-12

**Chapter 4: Examples: Dynamic Server Pages**

HCST and HCSP Example . . . . .	4-1
HCSF Example . . . . .	4-3
Common Code for Forms. . . . .	4-7
Retrieving File Information. . . . .	4-8
Referencing the File Extension . . . . .	4-8
Defining Form Information . . . . .	4-8
Defining Form Fields . . . . .	4-9
Defining Hidden Fields. . . . .	4-9
Submitting the Form . . . . .	4-9

**Appendix A: Third Party Licenses**

Apache Software License . . . . .	5-1
W3C® Software Notice and License . . . . .	5-2
Zlib License . . . . .	5-4
General BSD License. . . . .	5-5

General MIT License . . . . .	5-5
Unicode License . . . . .	5-6
Miscellaneous Attributions . . . . .	5-7



# ABOUT THIS GUIDE

This manual describes the details of *dynamic server pages*, which are files that are checked into the content server and then used to generate web pages dynamically. Dynamic server pages are used to customize the content server without the use of components.

This section covers these topics:

- ❖ [Audience](#) (page 1-1)
- ❖ [Conventions](#) (page 1-1)
- ❖ [Organization](#) (page 1-2)

## AUDIENCE

---

This guide is intended for developers who need to customize Content Server software to suit content management needs specific to a particular business or organization.





## CONVENTIONS

---

The following conventions are used throughout this guide:

- ❖ The notation `<Install_Dir>/` is used to refer to the location on your system where the content server instance is installed.
- ❖ Forward slashes (`/`) are used to separate the directory levels in a path name. A forward slash will always appear after the end of a directory name.

- ❖ Notes, technical tips, important notices, and cautions use these conventions:

Symbols	Description
	This is a note. It is used to bring special attention to information.
	This is a technical tip. It is used to identify information that can be used to make your tasks easier.
	This is an important notice. It is used to identify a required step or required information.
	This is a caution. It is used to identify information that might cause loss of data or serious system problems.

## ORGANIZATION

---

This manual includes the following sections:

- ❖ [Chapter 1 \(\*About This Guide\*\)](#) outlines the audience, organization, and conventions for this guide, and describes content management product distinctions and support options.
- ❖ [Chapter 2 \(\*Understanding Dynamic Server Pages\*\)](#) describes how to create and use dynamic server pages.
- ❖ [Chapter 3 \(\*HCSF Pages\*\)](#) describes the sections of an HCSF page in detail.
- ❖ [Chapter 4 \(\*Examples: Dynamic Server Pages\*\)](#) provides examples of dynamic server pages.
- ❖ The [Glossary](#) defines terms that are related to customizing the content server using dynamic server pages.

An index is provided at the end of this guide.



# UNDERSTANDING DYNAMIC SERVER PAGES

This section gives you an overview of the building blocks you'll need to create dynamic server pages. It includes the following sections:

- ❖ [About Dynamic Server Pages](#) (page 2-1)
- ❖ [Page Types](#) (page 2-3)
- ❖ [Creating Dynamic Server Pages](#) (page 2-4)
- ❖ [Syntax](#) (page 2-5)
- ❖ [Idoc Script Functions](#) (page 2-8)
- ❖ [HCSF Pages](#) (page 2-10)
- ❖ [Development Recommendations](#) (page 2-10)

## ABOUT DYNAMIC SERVER PAGES


---

Dynamic server pages are files that are checked into the content server and then used to generate web pages dynamically. Dynamic server pages include the following file formats:

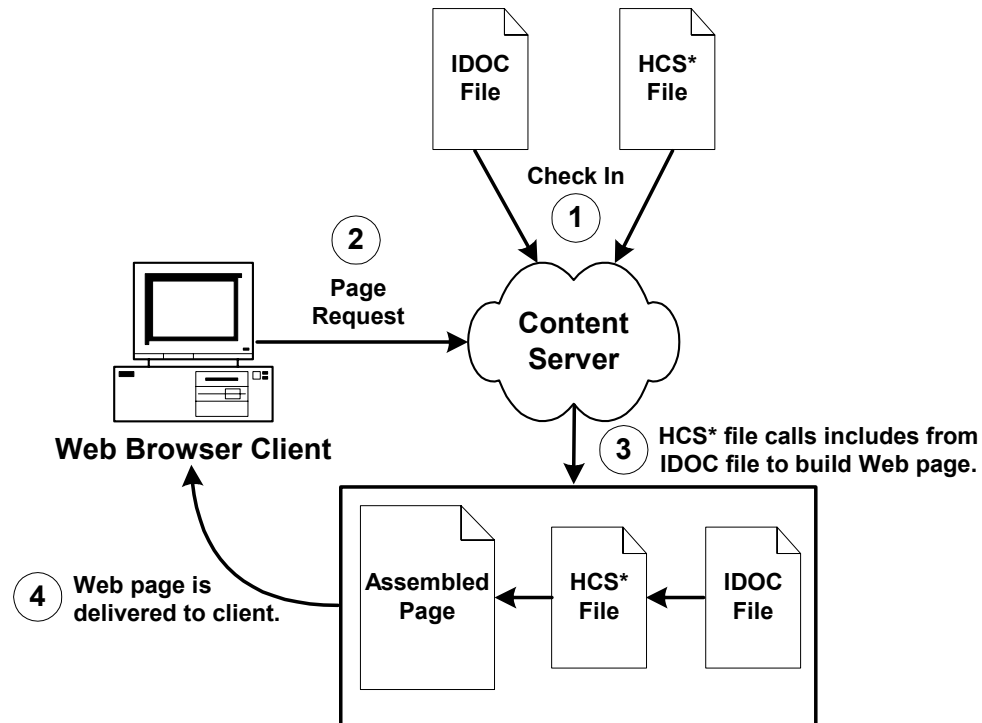
- ❖ IDOC
- ❖ HCST
- ❖ HCSP
- ❖ HCSF

When you use dynamic server pages, the content server assembles web pages dynamically using a custom template (HCST, HCSP, or HCSF file) that you have checked in to the content server. The template calls HTML includes from a text file (IDOC file) you have also checked in to the content server.

To make changes to the look-and-feel or navigation on a web page, you modify the HCS\* template page and/or the IDOC file, and then check in the revised files as new revisions. Your changes are available immediately.

 **Note:** See *Overview—Customization Methods—Dynamic Server Pages* in *Customizing Content Server* for an overview of the benefits and limitations of dynamic server pages.

**Figure 2-1** The dynamic server page process.



# PAGE TYPES

---

There are four types of dynamic server pages, which are identified in the content server by their four-character file name extensions:

- ❖ [IDOC File](#) (page 2-3)
- ❖ [HCST File](#) (page 2-3)
- ❖ [HCSP File](#) (page 2-3)
- ❖ [HCSF File](#) (page 2-4)



**Note:** See [Chapter 4 \(Examples: Dynamic Server Pages\)](#) for examples of each type of dynamic server page.

## IDOC File

---

An IDOC file is a text file containing HTML includes that are called by HCST, HCSP, and HCSF pages.



**Note:** See the *Using Components Guide* for detailed information on includes.

## HCST File

---

A Hypertext Content Server Template (HCST) file is a template page, similar to a standard content server template page, that is used as a framework for assembling a web page.

- ❖ HCST pages are typically used when the content of the page itself is dynamic or where Content Server functionality is needed, such as on a search page, search results page, or custom checkin page.
- ❖ Because this type of page consists mostly of dynamically assembled code, HCST files are not indexed in the content server.

## HCSP File

---

A Hypertext Content Server Page (HCSP) file is a published web page that displays actual web site content.

- ❖ HCSP files are typically created either by generating the web page through Content Publisher using an HCST page as a template, or by submittal of a form in the content server through an HCSF page.

- ❖ Because this type of page contains web-viewable content, HCSP files are indexed in the content server.

## HCSF File

---

A Hypertext Content Server Form (HCSF) file is similar to an HCSP file, except that it contains HTML form fields that can be filled out and submitted from a web browser.

- ❖ When a user fills out and submits a form from an HCSF page, an HCSP file is checked in as a separate content item with metadata defined by XML tags in the HCSF page.
- ❖ Because this type of page contains web-viewable content, HCSF files are indexed in the content server.



**Note:** See [HCSF Pages](#) (page 2-10) for more detail on HCSF pages.

## CREATING DYNAMIC SERVER PAGES

---

Although dynamic server pages are implemented in the content server differently than custom components, you will need to be familiar with Content Server component architecture concepts, particularly content server templates and HTML includes. For more information on these topics, see the *Content Server Using Components Guide*.

Use the following basic procedure to customize your content server using dynamic server pages:

1. Create an IDOC file with custom includes.
2. Check the IDOC file into the content server.
3. Create an HCST, HCSP, or HCSF file that references the IDOC file.
4. Check the HCS\* file into the content server.
5. Display the HCS\* file in your web browser by searching for it in the content server or linking to it from a published web page.



**Note:** Using dynamic server pages with Content Publisher can be a powerful tool for web publishing. See the Content Publisher documentation for more information.



**IDOC or HCST file:** <\$include MyIdocTag\$>

**HCSP or HCSF file:** <!--\$include MyIdocTag-->

In some situations, you may want to control the opening and closing of the HTML comment. In HCSP and HCSF files, this can be done by substituting other characters for the dash (-) in the closing tag of an Idoc Script expression.

For example:

```
<!--$a="ab"##> HTML comment remains open
<a href="<!--$myUrlAsVariable##">MyUrl</a> Static view does not see this
<!--$dummy=""--> <!--Ended the comment area-->.
```

In the above example, the pound sign (#) is substituted for the dash (-).

Another option in HCSP and HCSF files is to substitute brackets ([ ]) for the opening and closing tags (<>) in the standard HTML comment tags. This allows an XHTML parser to properly identify all the script when viewed statically, and causes Verity (if used as the search engine) to ignore the script block completely.

For example:

```
<!--$a="ab"--] HTML comment remains open
<a href="![!--$myUrlAsVariable--]">MyUrl</a> Static view does not see this
![!--$dummy=""--> <!--Ended the comment area-->.
```

## Comparison Operators

For HCSP and HCSF pages, the standard comparison operators (such as ==) cannot be used because of their special meaning to HTML parsers. Use the following comparison operators in dynamic server pages:

IDOC or HCST File	HCSP or HCSF File	Description
==	eq	Tests for equality.
!=	ne	Tests for inequality.
<	lt	Tests if less than.
>	gt	Test if greater than.
<=	le	Tests if less or equal than.
>=	ge	Tests if greater or equal than.

For example, the following code evaluates whether the variable *count* is greater than 10:

IDOC or HCST File	HCSP or HCSF File
<pre>&lt;\$if count &gt; 10\$&gt;   &lt;\$"Count is greater than"\$&gt; &lt;\$endif\$&gt;</pre>	<pre>&lt;!--\$if count gt 10--&gt;   &lt;!--"\$Count is greater than"--&gt; &lt;!--\$endif--&gt;</pre>

## Special Characters

For HCSP and HCSF pages, special characters such as the ampersand (&) cannot be used because of their special meaning to HTML parsers. You must use the standard HTML/XML escape format (such as `&amp;` or `&#038;`).

For example, in Idoc Script, a quotation mark can be included in a string by preceding it with a backslash escape character. However, in an HCSP or HCSF page, the quotation mark character must be represented by an HTML escape character:

**IDOC or HCST file:** "Enter \"None\" in this field."

**HCSP or HCSF file:** "Enter &quot;None&quot; in this field."

In an HCST page, a line feed is inserted using `\n`. In an HCSP page, insert the line feed directly in the file or encode it in the XML using the numeric ASCII number for a line feed.



**Note:** It is especially important to use the `&amp;` escape character when you call the `docLoadResourceIncludes` function from an HCSP or HCSF page. See [docLoadResourceIncludes Function](#) (page 2-8).



**Tech Tip:** You can now substitute the word *join* for the `&` string join operator. For example, you can write `[!-$a join b-]` instead of `[!-$a & b-]`. The first is accepted by an XML parser inside an attribute of a tag, but the second is not.

## Referencing Metadata

For dynamic server pages, several metadata values are stored with a *ref:* prefix, which makes them available to the page but does not replace ResultSet values. (This prevents “pollution” of ResultSets by dynamic server pages.)

When you reference any of the following metadata values on a dynamic server page, you must include the *ref:* prefix:

- ❖ hasDocInfo
- ❖ dDocName
- ❖ dExtension
- ❖ dSecurityGroup
- ❖ isLatestRevision
- ❖ dDocType

For example, the following statement determines if the document type is *Page*:

```
<$if strEquals(ref:dDocType,"Page")>
```

## IDOC SCRIPT FUNCTIONS

---

Two special Idoc Script functions are required for dynamic server pages:

- ❖ [docLoadResourceIncludes Function](#) (page 2-8)
- ❖ [executeService Function](#) (page 2-9)

### docLoadResourceIncludes Function

---

To be able to use the HTML includes in an IDOC file, an HCS\* file must call the *docLoadResourceIncludes* function. This function loads all the includes from the specified IDOC file for use in assembling the current page.

For example:

**HCST file:** `<$docLoadResourceIncludes("dDocName=system_std_page&RevisionSelectionMethod=Latest")>`

**HCSP or HCSF file:** `<!--$docLoadResourceIncludes("dDocName=system_std_page&RevisionSelectionMethod=Latest")-->`

- ❖ The native file for the specified content item must have an *.idoc* extension.
- ❖ The *docLoadResourceIncludes* call must be placed before the first include call in the HCS\* file. It is recommended that you place this function within the <HEAD> section of the page.
- ❖ You must use the correct ampersand character when you call the *docLoadResourceIncludes* function from an HCS\* page. See [Special Characters](#) (page 2-7).



## Parameters

Use the following parameters with the *docLoadResourceIncludes* function to specify which IDOC file to call.

- ❖ You must define either a *dDocName* or a *dID*; do not use both parameters together.
- ❖ If you define a *dDocName*, you must define *RevisionSelectionMethod* to be *Latest* or *LatestReleased*.
- ❖ If you define a *dID*, do not define a *RevisionSelectionMethod*, or define the *RevisionSelectionMethod* to be *Specific*.

Parameter	Description
dDocName	Specifies the Content ID of the IDOC file. This parameter should always be present when the Content ID is known. Error messages assume that it is present, as do other features such as forms.
dID	Specifies the unique ID number of a particular revision of the IDOC file.
RevisionSelectionMethod	Specifies which revision of the IDOC file to use. <b>Latest</b> —The latest checked in revision of the document is used (including revisions in a workflow). <b>LatestReleased</b> —The latest released revision of the document is used. <b>Specific</b> —Use only with <i>dID</i> .
Rendition	Specifies which rendition of the IDOC file to use. <b>Primary</b> —The primary (native) file. This is the default if no <i>Rendition</i> is specified. <b>Web</b> —The web-viewable file. <b>Alternate</b> —The alternate file.

## executeService Function

The *executeService* function executes a content server service from within a dynamic server page. For example:

**HCST file:** <\$executeService("GET\_SEARCH\_RESULTS")\$>

**HCSP or HCSF file:** <!--\$executeService("GET\_SEARCH\_RESULTS")-->

- ❖ Services that can be called with the *executeService* function must be “scriptable”, meaning that they do not require parameter input.
- ❖ Scriptable services have an access level of 32 or more. See the *Services Reference Guide* for more information.
- ❖ See the `<install_dir>/shared/resources/std_services.htm` file for a list of standard content server services.
- ❖ See the *Idoc Script Reference Guide* for more information on the *executeService* function.
- ❖ See the *Services Reference Guide* for more information on services.



**Caution:** Use services sparingly. Too many service calls on a page can affect performance and limit scalability.

## HCSF PAGES

---

In addition to following the standard formatting rules for content server templates and HTML forms, HCSF pages require a number of special sections and tags that allow the content server to process them. See [Chapter 3 \(HCSF Pages\)](#) for more information.



**Note:** See [HCSF Example](#) (page 4-3) for an example of a complete HCSF page.

## DEVELOPMENT RECOMMENDATIONS

---

This section provides some guidelines to assist you in developing dynamic server pages. It includes the following sections:

- ❖ [General Tips](#) (page 2-10)
- ❖ [HCSF Tips](#) (page 2-11)

### General Tips

---

The following recommendations apply to the development of all types of dynamic server pages:

- ❖ Keep templates as simple and free of code as possible. Strive to have only HTML includes in your templates, with all code and conditionals in an IDOC file. This is

especially helpful for HCSF pages, where submitted forms will also reflect changes made to the IDOC file.

- ❖ Whenever you are customizing the content server, you should isolate your development efforts from your production system. Keep in mind that frequent revisions to dynamic server pages can result in a large number of obsolete content items. You should do as much work on a development system as possible before deploying to a production instance, and you may need to delete out-of-date pages regularly.
- ❖ When you develop a web site using dynamic server pages, think of the development and contribution processes in terms of *ownership*:
  - Structure, including site design and navigation, is owned by the webmaster. When you use dynamic server pages, structure is contained in and controlled with includes that are defined in IDOC files.
  - Content—the actual text of the web pages—is owned by the contributors. When you use dynamic server pages, content is contained primarily in HCSP files that make use of the includes in the IDOC files.
- ❖ Using dynamic server pages with Content Publisher can be a powerful tool for web publishing. You can create content using Word documents or HCSF pages, and then use Content Publisher to convert the documents to published HCSP files. You can also use the “include before” and the “include after” options in the SCP template to insert additional Idoc Script includes.
- ❖ If you publish dynamic server pages with Content Publisher, use the prefix option for easy identification of your documents.
- ❖ Use a consistent naming convention. For example, for “system” level includes, you could name your IDOC file *system\_std\_page*, and then name each include in that file with the prefix *system\_*. This makes locating the includes easier.
- ❖ You may want to create a content type for each type of dynamic server page (such as *HCSF\_templates* or *submitted\_forms*).
- ❖ In accordance with good coding practices, you should always put comments in dynamic server pages to document your customizations.

## HCSF Tips

---

The following recommendations apply specifically to the development of HCSF pages:

- ❖ When designing a form, consider how the template will be used:

- Will this template change depending on the role of the user submitting the form?
  - Will the submitted content enter into a criteria workflow?
  - What default metadata values should be set?
  - Does the form contain ResultSets for multiple line entries?
- ❖ To see the form parameters as they are passed from the web browser to the web server, filtered through the content server, and then passed back to the web browser, change the METHOD attribute in the include code from a POST to a GET:
- ```
<form name="<$formName$>" method="GET" action="<$HttpCgiPath$>">
```
- ❖ If you add a form field called DataScript to a form being submitted, then any Idoc Script for that value will be evaluated by content server when the form is processed by content server.

# HCSF PAGES

In addition to following the standard formatting rules for content server templates and HTML forms, HCSF pages require a number of special sections and tags that allow the content server to process them. These special sections appear in the following order in a typical HCSF file:

- ❖ [Load Section](#) (page 3-1)
- ❖ [Data Section](#) (page 3-3)
- ❖ [Form Section](#) (page 3-10)



**Note:** See [HCSF Example](#) (page 4-3) for an example of a complete HCSF page.

## LOAD SECTION

---

The load section at the beginning of an HCSF page declares the file as an HTML file, loads an IDOC file, and loads other information about the page. The following is a typical load section:

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML//EN">
<html>
<head>
<!--$docLoadResourceIncludes("dDocName=my_idoc_page&
RevisionSelectionMethod=Latest")-->
<meta NAME="idctype" CONTENT="form; version=1.0">
<!--$defaultPageTitle="Department News Form"-->
<!--$include std_html_head_declarations-->
</head>
```

The load section includes the following:

- ❖ [HTML Declaration](#) (page 3-2)
- ❖ [docLoadResourceIncludes Function](#) (page 3-2)
- ❖ [Meta Tag](#) (page 3-2)
- ❖ [Variables and Includes](#) (page 3-2)

## HTML Declaration

---

The HTML declaration identifies the file as an HTML file using the following syntax:

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML//EN">
```

## docLoadResourceIncludes Function

---

The docLoad ResourceIncludes function loads all the includes from the specified IDOC file for use in assembling the current page. See [docLoadResourceIncludes Function](#) (page 2-8) for more information.

## Meta Tag

---

The *meta* tag is used by Content Publisher to identify that this is a special type of page.

- ❖ This tag is not required if the form is not being published through Content Publisher.
- ❖ The *meta* tag must be placed inside the <HEAD> section of your HTML file.
- ❖ Use the following syntax for the *meta* tag:

```
<meta NAME="idctype" CONTENT="form; version=1.0">
```

## Variables and Includes

---

The <HEAD> section of your HCSF page can contain variable definitions and HTML includes as necessary. For example, the following lines define the default page title and load the *std\_html\_head\_declarations* code:

```
!--$defaultPageTitle="Department News Form"-->
<!--$include std_html_head_declarations-->
```

## DATA SECTION

---

The data section contains rules and metadata information that is used to process the form. There is a close relationship between the information in the data section and the presentation of the page:

- ❖ Upon delivery of the HCSF page to the user, the information in the data section is parsed into a `DataBinder` and merged into the [Form Section](#) (page 3-10).
- ❖ Upon form submittal, the information in the data section is merged with the request and written out again to the data section.



**Note:** See *DataBinder* and *ResultSet Section* in the *Component Architecture Guide* for more information.

This section covers these topics:

- ❖ [Data Section Structure](#) (page 3-3)
- ❖ [idcformrules Tag](#) (page 3-4)
- ❖ [Metadata Tags](#) (page 3-5)
- ❖ [Nested Tags](#) (page 3-5)
- ❖ [Referencing XML Tags](#) (page 3-6)
- ❖ [Form Elements](#) (page 3-6)
- ❖ [ResultSets](#) (page 3-7)

## Data Section Structure

---

The data section consists of XML tags that are placed between *idcbegindata* and *idcenddata* Idoc Script tags. For example:

```
<!--$idcbegindata-->
  <idcformrules isFormFinished="0"/>
  <model_number content="html">AB-123</model_number>
  <revision>12</revision>
  ...
<!--$idcenddata-->
```

- ❖ The data section must be placed inside the `<BODY>` section of your HTML file, before the beginning of the form section.

- ❖ You can place Idoc Script variable definitions and includes before or after the data section, but not within it.
- ❖ Two types of XML tags are used in the data section:
  - [idcformrules Tag](#) (page 3-4)
  - [Metadata Tags](#) (page 3-5)
- ❖ You can also use the following types of formatting in the data section:
  - [Nested Tags](#) (page 3-5)
  - [Referencing XML Tags](#) (page 3-6)
  - [Form Elements](#) (page 3-6)
  - [ResultSets](#) (page 3-7)

## idcformrules Tag

---

The *idcformrules* tag defines content server-specific rules in the data section. This tag requires one attribute, either the [isFormFinished Attribute](#) (page 3-4) or [resultsets Attribute](#) (page 3-4).

### isFormFinished Attribute

The *isFormFinished* attribute indicates whether the form can be submitted again or not.

- ❖ Use the following format to specify that the form can be submitted again:
 

```
<idcformrules isFormFinished="0"/>
```
- ❖ Use the following format to specify that the form cannot be submitted again. This will result in a read-only form:
 

```
<idcformrules isFormFinished="1"/>
```

### resultsets Attribute

The *resultsets* attribute indicates which XML tags in the data section will be interpreted as ResultSets.

- ❖ This attribute specifies one or more XML tag names separated by commas. For example:
 

```
<idcformrules resultsets="volume,chapter">
```



- ❖ During delivery of an HCSF page to the user, the content server reads the *resultsets* attribute and, if necessary, places empty ResultSets with the specified names into the DataBinder so they are available for merging.
- ❖ See [ResultSets](#) (page 3-7) for more information on ResultSet formatting in the data section.

## Metadata Tags

---

Metadata tags specify the metadata values that appear in the form fields when the form is displayed in a browser. For example:

```
<model_number>AB-123</model_number>
```

### content Attribute

Each metadata tag can be assigned a *content* attribute that indicates which type of content the tag contains. For example:

```
<model_number content="html">AB-123</model_number>
```

- ❖ The value of the *content* attribute can be either *html* or *text*:
  - **text** indicates that the content of the tag should be interpreted strictly as text.
  - **html** indicates that the content of the tag should be interpreted as HTML code.
- ❖ If the *content* attribute is not specified for a metadata tag, it defaults to *html*.
- ❖ Content Publisher ignores all other attributes except the *content* attribute.

## Nested Tags

---

If you are not publishing HCSF pages through Content Publisher, you can use nested XML tags (also called *nodes*) within the data section. In the following example, the `<section>` tag is nested in the `<chapter>` tag:

```
<chapter title="Chapter 1">
  This is the beginning of the chapter.
  <section title="First Section">
    This is the first section of the chapter.
  </section>
</chapter>
```



**Note:** Nested XML tags are not allowed in Content Publisher.

## Referencing XML Tags

---

- ❖ To refer to a nested tag, start with the root-level tag and use an exclamation point (!) between tag levels. For example:  
`chapter!section`
- ❖ To refer to the attribute of any tag, use a colon (:) after the tag name. For example:  
`chapter!section:title`
- ❖ If you reference a tag in the data section, the tag value can be merged back into the data section upon form submission only if one of the following are true:
  - The root tag has already been referenced in the data area.
  - The root tag is referenced in an *ExtraRootNodes* form element.
  - A prefix part of the tag is referenced as a ResultSet in the *resultsets* form element.
- ❖ Default values can be specified by applying the *:default* suffix to a tag path. Note that default elements may contain Idoc Script for further evaluation. For example, to specify a default *dDocTitle*:  
`<input type=hidden name="dDocTitle:default" value="<$'MyTitle ' & dateCurrent()$">>0`

## Form Elements

---

- ❖ The *ExtraRootNodes* form element enables you to add tags by creating an Idoc Script variable and then appending the tag names to it, rather than specifying the tags in the data section of the form. At the end of your form, you can substitute a string value in place of the *ExtraRootNodes* value to be merged back into the data section.
- ❖ The *resultsets* form element enables you to add a tag as a ResultSet, rather than specifying the ResultSet in the data section.
- ❖ Both the *ExtraRootNodes* and *resultset* form elements take a comma-separated list of tags.
- ❖ For example, the following form elements add the `mychapters!chapter` tag as a valid ResultSet if it is not already defined in the *idcformrules resultsets* attribute. It also adds, if necessary, the root tag `mychapters`.  
`<input type=hidden name="resultsets" value="mychapters!chapter">`  
`<input type=hidden name="ExtraRootNodes" value="mychapters">`

## ResultSets

---

You can define a ResultSet using XML tags within the data section.

- ❖ You must use the [resultsets Attribute](#) (page 3-4) of the *idcformrules* tag to specify a ResultSet.
- ❖ The tags must be completely qualified and the full reference path from the root node must be used.
- ❖ The columns in the ResultSet are the tag content and the tag attributes.
- ❖ See [Repeated Tags in a ResultSet](#) (page 3-8) and [Nested Tags in a ResultSet](#) (page 3-8) for limitations on repeating and nesting XML tags in a ResultSet.

### Example

In the following example, two ResultSets named *volume* and *chapter* are defined by XML tags:

```
<idcformrules resultsets="volume,chapter">
  <volume title="First Volume">
    Volume content here
  </volume>
  <chapter title="First Chapter">
    Chapter content here
  </chapter>
```

This evaluates into two ResultSets with two columns each:

```
@ResultSet volume
2
volume
volume:title
Volume content here
First Volume
@end
@ResultSet chapter
2
chapter
chapter:title
Chapter content here
First Chapter
@end
```

## Repeated Tags in a ResultSet

If you are not publishing HCSF pages through Content Publisher, you can use repeated tags within a ResultSet in the data section. Repeated tags are typically useful for looping over code to create the ResultSet.

- ❖ Repeated tags are not allowed unless they are part of a ResultSet.
- ❖ Repeated XML tags are not allowed in Content Publisher.
- ❖ In the following example, the `chapter` tag is repeated in the *chapter* ResultSet:

```
<idcformrules resultsets="chapter">
<chapter title="First Chapter">
    Some content here
</chapter>
<chapter title="Second Chapter">
    More content here
</chapter>
```

This evaluates into a ResultSet with two columns and two rows:

```
@ResultSet chapter
2
chapter
chapter:title
Some content here
First Chapter
More content here
Second Chapter
@end
```

## Nested Tags in a ResultSet

A ResultSet can have nested tags, but the nested tags may not be repeated within a parent tag. For example, an additional `<section>` tag would not be allowed within the first

`<chapter>` tag:

```
<idcformrules resultsets="chapter">
<chapter title="First Chapter">
    Some content here
    <section title="First Section of First Chapter">
        Section content
    </section>
</chapter>
<chapter title="Second Chapter">
    More content here
</chapter>
```

This evaluates into a `ResultSet` with four columns and four rows (the last two cells are blank):

```
@ResultSet chapter
4
chapter
chapter:title
chapter!section
chapter!section:title
Some content here
First Chapter
Section Content
First Section of First Chapter
More content here
Second Chapter

@end
```

## Editing a `ResultSet`

- ❖ Updating a specific field in a `ResultSet` requires that you indicate the `ResultSet` row number in the request parameter. The `#` character is used by the Content Server to indicate a specific row. If you do not specify a row with the `#` character, then a row is appended. If you specify a row `#` that does not yet exist, then empty rows are added sufficiently to provide a row to be edited.

For example, to update the first row (row 0) of the `ResultSet`, you might use the following code:

```
<input type="text" name="comment#0"
  value="new comment">
<input type="text" name="comment!title#0"
  value="new title"
```

- ❖ Insert new fields into a `ResultSet` by using the exclamation point character (!). For example, to insert author and title fields into the `comment` `ResultSet`, name the input fields `comment!author` and `comment!title`. If those fields are not in the `ResultSet`, they will be added when the form is submitted.
- ❖ To delete a row in a `ResultSet`, empty all the values so they are blank. For example, to delete the first row entirely:

```
<input type="hidden" name="comment#0" value="">
<input type="hidden" name="comment!title#0" value="">
<input type="hidden" name="comment!date#0" value="">
<input type="hidden" name="comment!author#0" value="">
```

Another method for deleting rows from a ResultSet is to set the *DeleteRows* form element to a list of comma-separated pairs of ResultSet name and row number. For example, if you want to delete row 2 from the *comment* ResultSet and row 5 from the *book* ResultSet, then the *DeleteRows* form element would be set to the following comma-separated pairs:

```
comment : 2 , book : 5 .
```

## FORM SECTION

---

The form section contains the code for presentation of the HTML form elements and any other functionality that the page requires. The form properties, form fields, and form buttons are placed in an HTML table to control the formatting of the assembled web page.

This section covers these topics:

- ❖ [Form Begin](#) (page 3-10)
- ❖ [Form Properties](#) (page 3-11)
- ❖ [Form Fields](#) (page 3-11)
- ❖ [Form Buttons](#) (page 3-12)
- ❖ [Form End](#) (page 3-12)



**Note:** See [Common Code for Forms](#) (page 4-7) for additional code examples.

### Form Begin

---

The form section begins with the following Idoc Script:

```
<!--$formName="HTMLForm"-->
<!--$include std_html_form_submit_start-->
```

The *std\_html\_form\_submit\_start* include in the *std\_page.htm* resource file contains the following code, which creates a standard HTML form using a POST method, sets the *IdcService* to *SUBMIT\_HTML\_FORM*, and sets the *dID* variable to the value of the current HCSF page:

```
<form name="<$formName$>" method="POST"action="<$HttpCgiPath$>">7
<input type=hidden name="IdcService" value="SUBMIT_HTML_FORM">
<input type=hidden name="dID" value="<$SourceID$>">
```

## Form Properties

---

The form table typically begins with the following property definitions, which create the fields as form fields, allow the fields to be edited, and set the size of the field caption area:

```
<!--$isFormSubmit=1,isEditMode=1-->
<!--$captionFieldWidth=200, captionEntryWidth=80-->
```

## Form Fields

---

The following lines are typically used to create each input field:

```
<!--$eval("<$product_name:maxLength=250$>")-->
<!--$fieldName="model", fieldCaption="Model Number"-->
<!--$include std_display_field-->
```



**Tech Tip:** Some fields may require additional code for proper display. For example, you might need to override the standard *std\_memo\_entry* include to increase the size of text areas. You can do this by defining a custom include in the IDOC file:

```
<@dynamicalhtml std_memo_entry@>
  <textarea name="<$fieldName$>" rows=15 cols=50
  wrap=virtual><$fieldValue$></textarea>
<@end@>
```

## DataScript

If you add a form field called DataScript to a form being submitted, then any Idoc Script for that value will be evaluated by content server when the form is processed by content server.

### Example

There are two tables (coming from the data island inside the hcsp form) with an entry in one that references entries in another. Your goal is to change a value in a specific column and row in the second table when you update a row in the first table. To accomplish this, you can write javascript to set the DataScript value with Idoc script:

```
modifyRowAndColumn(row, column, value)
{
document.myform.DataScript = "<$setValue('#local', 'table2!'"+ column + "'"+ row
+
"', '" + value + "')$>";
}
```

Then, when you call the function with `column = "myColumn"` and `row="1"` and `value = "Test"` while submitting the update form, the resulting DataScript value before submit would be the following:

```
DataScript.value = <$setValue('#local', 'table2!myColumn#1', 'Test')$>
```

The result would be the column `table2!myColumn` in row 1 of the table `table2` would be updated with the value `Test` after the form was submitted.

Another way of saying this is that the DataScript can allow arbitrary edits of other entries in the data island without having to actually create html form fields that reference their names.

## Form Buttons

---

The following lines are typically used to create the form submission and Reset buttons:

```
<input type=submit name=Submit value=" Submit ">  
<input type=reset name=Reset value="Reset">
```

## Form End

---

After all the form elements and default values have been defined, the form must end with a `</form>` tag.



# EXAMPLES: DYNAMIC SERVER PAGES

This section presents examples that show how the dynamic server pages work together to modify content server behavior. It includes the following sections:

- ❖ [HCST and HCSP Example](#) (page 4-1)
- ❖ [HCSF Example](#) (page 4-3)
- ❖ [Common Code for Forms](#) (page 4-7)

## HCST AND HCSP EXAMPLE

---

This example shows you how to create a simple HCST page and HCSP page:

1. Create an IDOC file with a custom include.

**Figure 4-2** Custom include

```
<@dynamichtml HelloWorld@>  
<H1>Hello World</H1>  
<@end@>
```

This include is named *HelloWorld*.

This include defines one line of HTML code.

2. Save the file as *helloworld.idoc*.
3. Check the IDOC file into the content server with a Content ID of *helloworld*. The IDOC file is now available to any HCS\* pages that reference it.
4. Create an HCST file that references the HelloWorld include:

Figure 4-3 HCST file referencing custom include

```
<HTML>
<HEAD>
<docLoadResourceIncludes ("dDocName=helloworld&RevisionSelectionMethod=LatestReleased")$>
</HEAD>
<BODY>
You should see it:
<include HelloWorld$>
</BODY>
</HTML>
```

This line loads the *helloworld.idoc* file so the includes in the IDOC file are available to this page.

The second line displays the code from the *HelloWorld* include in the *helloworld.idoc* file. Note the use of standard IDoc Script tags <\$...\$>.

5. Save the file as *helloworld.hcst*.
6. Check the HCST file into the content server.
7. Create an HCSP file that references the HelloWorld include:

Figure 4-4 HCSP file referencing custom include

```
<HTML>
<HEAD>
<!--$docLoadResourceIncludes ("dDocName=helloworld&RevisionSelectionMethod=LatestReleased")-->
</HEAD>
<BODY>
You should see it:
<!--$include HelloWorld-->
</BODY>
</HTML>
```

This line loads the *helloworld.idoc* file so the includes in the IDOC file are available to this page.

The second line displays the code from the *HelloWorld* include in the *helloworld.idoc* file. Note the use of HTML comment tags <!--...-->.

8. Save the file as *helloworld.hcsp*.
9. Check the HCSP file into the content server.
10. Search for the *helloworld* content items in the content server.

11. Display the HCST file and HCSP files in your web browser. They should both look like this:

Figure 4-5 *HelloWorld* content item displayed in a web browser.



## HCSF EXAMPLE

---

This example shows you a typical HCSF page and its associated IDOC file. This example creates a form that users can fill out and submit to enter product descriptions as content items.

1. Create an HCSF file that references an IDOC file named *form\_std\_page*:

**Figure 4-6** Product description form HCSF file.

```

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML//EN">
<html>
<!--$docLoadResourceIncludes("dDocName=form_std_page&amp;
RevisionSelectionMethod=Latest")-->
<head>
<meta NAME="idctype" CONTENT="form; version=1.0">
<!--$include form_head_section-->
</head>
<!--$include form_pre_xml_section-->
<!--$idcbegindata-->
  <idcformrules isFormFinished="0"/>
  <product_name content="html">
    </product_name>
  <model_number content="html">
    SC-
  </model_number>
  <summary_description>
    Enter a summary here.
  </summary_description>
  <full_description>
    Enter a full description here.
  </full_description>
  <author>
    ProdMgr67
  </author>
  <division>
    Household Products
  </division>
  <revision>
  </revision>
<!--$idcenddata-->
<!--$include form_post_xml_section-->
</body>
</html>

```

This line loads the IDOC file with the Content ID of *form\_std\_page* so the includes in the IDOC file are available to this page.

The *meta* tag is required for Content Publisher to acknowledge that this is a Content Server HTML form.

These includes, defined in the *form\_std\_page* IDOC file, generate the code at the beginning of the web page.

The *isFormFinished* attribute of the *idcformrules* tag tells the content server that the form is not finished, so the fields can be edited and the form can be submitted.

The *content* property does not have to be set for each tag; it defaults to *html*.

The *idcbegindata* and *idcenddata* tags define the XML tagged area, which specifies rules and initial metadata values for the form.

The text in each set of XML tags will populate the corresponding field on the form.

This include, defined in the *form\_std\_page* IDOC file, generates the code at the end of the web page.

2. Save the file as *product\_form.hcsf*.
3. Check the HCSF file into the content server.
4. Create an IDOC file with custom includes:

**Figure 4-7** IDOC file with custom includes

```

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML//EN">
<html>
<body>

<@dynamichtml form_head_section@>
<!--standard includes for a standard hcsp page-->
<$defaultPageTitle="Product Description Form"$>
<$include std_html_head_declarations$>
<@end@>

<@dynamichtml form_pre_xml_section@>
<!--This code is here for static viewing.-->
<$if 0$>
  <body>
<$endif$>

<$include body_def$>
<@end@>

<@dynamichtml form_post_xml_section@>
<$include std_page_begin$>
<$include std_header$>

<$formName="HTMLForm"$>
<$include std_html_form_submit_start$>

<table>

<$if (strEquals(ref:dExtension,"hcsf"))$>
  <$isHcsf=1$>
<$else$>
  <$isHcsp=1$>
<$endif$>

<$if isHcsf$>
  <$isFormSubmit=1,isEditMode=1$>
<$endif$>

<$captionFieldWidth=150,captionEntryWidth=200$>
  
```

The *form\_head\_section* include defines the page title and the code for the standard HTML head section (referencing the *std\_html\_head\_declarations* include in the *std\_page.htm* resource file).

The *form\_pre\_xml\_section* include allows the page to be viewed statically and defines code for a standard content server web page (referencing the *body\_def* include in the *std\_page.htm* resource file).

These includes, which are defined in the *std\_page.htm* resource file, define code for a standard content server web page.

These lines define the form name and the code for a standard HTML form (referencing the *std\_html\_form\_submit\_start* include in the *std\_page.htm* file).

This conditional determines if this is an editable form or a page that has already been submitted, based on the file name extension.

If this is an editable page (*isHcsf=1*), this conditional sets variables that create the fields as form fields and allow the fields to be edited.

The *form\_post\_xml\_section* include defines the form fields.

This line sets the width of the table cells for field captions to 150 pixels and sets the width of the table cells for input fields to 200 pixels.

```

<$eval("<$product_name:maxLength=250$>")$>
<$fieldName="product_name", fieldCaption="Product Name"$>
<$if isHcsp$><$isInfoOnly=1$><$endif$>
<$include std_display_field$>

<$eval("<$model_number:maxLength=250$>")$>
<$fieldName="model_number", fieldCaption="Model Number"$>
<$if isHcsp$><$isInfoOnly=1$><$endif$>
<$include std_display_field$>

<$fieldName="summary_description",
  fieldCaption="Summary Description",
  fieldType="Memo"$>
<$if isHcsp$><$isInfoOnly=1$><$endif$>
<$include std_display_field$>

<$fieldName="full_description",
  fieldCaption="Full Description",
  fieldType="Memo"$>
<$if isHcsp$><$isInfoOnly=1$><$endif$>
<$include std_display_field$>

<$eval("<$author:maxLength=250$>")$>
<$fieldName="author", fieldCaption="Author"$>
<$if isHcsp$><$isInfoOnly=1$><$endif$>
<$include std_display_field$>

<$eval("<$division:maxLength=250$>")$>
<$fieldName="division", fieldCaption="Division"$>
<$if isHcsp$><$isInfoOnly=1$><$endif$>
<$include std_display_field$>

<$eval("<$revision:maxLength=250$>")$>
<$fieldName="revision", fieldCaption="Revision"$>
<$if isHcsp$><$isInfoOnly=1$><$endif$>
<$include std_display_field$>
</tr>
<tr>
  <td colspan=2><hr></td>
</tr>
<tr align=center>
  <td colspan=2>
    <$if isHcsf$>
      <input type=submit name=Submit value=" Submit ">
      <input type=reset name=Reset value="Reset">
    <$endif$>

    <input type=hidden name="dDocTitle:default"
      value="<$'Product Description ' & dateCurrent ()$>">
  </td>
</tr>
</table>
</form>
<$include std_page_end$>
<@end@>
</body>
</html>

```

This section defines the form fields that appear on the web page.

The eval function sets the maximum length of a text field to 250 characters

This tag defines the name, caption, and type of field. If the fieldType is not defined, it defaults to Text.

If this is a form that has already been submitted (isHcsp=1), this conditional sets a variable that makes the form field read-only.

The std\_display\_field include, defined in the std\_page.htm resource file, defines code that creates the form field.

If this is an editable form (isHcsf=1), this conditional creates the Submit and Reset buttons.

This line generates the document title (dDocTitle) automatically.

The std\_page\_end include, defined in the std\_page.htm resource file, generates the code at the end of the web page.

5. Save the file as *form\_std\_page.idoc*.
6. Check the IDOC file into the content server with a Content ID of *form\_std\_page*. (This is the name that is referenced by the HCSF page.)
7. Search for the HCSF content item in the content server.
8. Click the link to display the HCSF page in your web browser. It should look like this:

**Figure 4-8** Sample form displayed in a web browser.

Product Name	<input type="text"/>
Model Number	<input type="text" value="SC-"/>
Summary Description	<input type="text" value="Enter a summary here."/>
Full Description	<input type="text" value="Enter a full description here."/>
Author	<input type="text" value="prod_mgr_001"/>
Division	<input type="text" value="Household Products"/>
Revision	<input type="text"/>

9. Fill out the form with some sample values and click **Submit**.

A content item is created as an HCSP page.

10. Search for the HCSP page in the content server.

11. Click the link to display the HCSP page in your web browser. It should look like this:

**Figure 4-9** Link displaying HCSP page

Product Name:	Super Cleaner
Model Number:	SC-1
Summary Description:	This product cleans everything! You can use Super Cleaner in the kitchen, bath, laundry, garage--anywhere there's dirt.
Full Description:	kitchen, bath, laundry, garage--anywhere there's dirt.
Author:	ProdMgr67
Division:	Household Products
Revision:	New

## COMMON CODE FOR FORMS

---

This section describes some of the features that are commonly used in HCSF pages and associated IDOC files. The following are included:

- ❖ [Retrieving File Information](#) (page 4-8)

- ❖ [Referencing the File Extension](#) (page 4-8)
- ❖ [Defining Form Information](#) (page 4-8)
- ❖ [Defining Form Fields](#) (page 4-9)
- ❖ [Defining Hidden Fields](#) (page 4-9)
- ❖ [Submitting the Form](#) (page 4-9)

## Retrieving File Information

---

Executing the service DOC\_INFO\_SIMPLE makes metadata from a specific file available to the page. For example:

```
<$dID=SourceID$>  
<$executeService("DOC_INFO_SIMPLE")$>
```

## Referencing the File Extension

---

Use the following statement to determine whether the form is submitted (hcsf) or unsubmitted (hcsp):

```
<$if (strEquals(ref:dExtension,"hcsf"))$>  
  <$isHcsf=1$>  
<$else$>  
  <$isHcsp=1$>  
<$endif$>
```



**Note:** See [Referencing Metadata](#) (page 2-7) for information on the *ref:* prefix.

## Defining Form Information

---

The following code defines the form name and the standard include to start an HTML form:

```
<$formName="HTMLForm"$>  
<$include std_html_form_submit_start$>
```

The following is typical code that defines form properties:

```
<table border=0 width=100%>  
<$isEditMode=1,isFormSubmit=1$>  
<$captionFieldWidth="25%", captionEntryWidth="75%"$>
```



## Defining Form Fields

---

Use standard Idoc Script variables and the *std\_display\_field* include to display the form fields. For example:

```
<${fieldName="news_author",fieldDefault=dUser,fieldCaption=
"Author",isRequired=1,requiredMsg = "Please specify the author."$>
<$include std_display_field$>
```

Some fields might require extra code to display the field correctly. For example, the standard text area for a memo field is 3 rows by 40 columns, but you might need to override the standard include to increase the size of the text area:

### ***Standard std\_memo\_entry Include***

```
<@dynamichtml std_memo_entry@>
  <textarea name="<${fieldName$}" rows=3 cols=40 wrap=virtual>
    <${fieldValue$}></textarea>
<@@end@>
```

### ***Custom std\_memo\_entry Include***

```
<@dynamichtml std_memo_entry@>
  <textarea name="<${fieldName$}" rows=15 cols=50 wrap=virtual>
    <${fieldValue$}></textarea>
<@@end@>
```

## Defining Hidden Fields

---

You can specify metadata for a submitted form (hcomp) by defining a hidden field, which contributors cannot change. For example, use the following code to assign the document type *News\_Forms* to each submitted form:

```
<input type=hidden name="dDocType" value="News_Forms">
```

To specify the security group of the submitted forms:

```
<input type=hidden name="dSecurityGroup" value="Public">
```

## Submitting the Form

---

When a form is submitted, you may want to call a Java function to perform additional validation or processing. For example:

```
<input type=button name=Submit value="Save" onClick="postCheckIn(this.form)">
```





# THIRD PARTY LICENSES

## OVERVIEW

---

This appendix includes a description of the Third Party Licenses for all the third party products included with this product.

- ❖ [Apache Software License](#) (page 5-1)
- ❖ [W3C® Software Notice and License](#) (page 5-2)
- ❖ [Zlib License](#) (page 5-4)
- ❖ [General BSD License](#) (page 5-5)
- ❖ [General MIT License](#) (page 5-5)
- ❖ [Unicode License](#) (page 5-6)
- ❖ [Miscellaneous Attributions](#) (page 5-7)

## APACHE SOFTWARE LICENSE

---

```
* Copyright 1999-2004 The Apache Software Foundation.  
* Licensed under the Apache License, Version 2.0 (the "License");  
* you may not use this file except in compliance with the License.  
* You may obtain a copy of the License at  
* http://www.apache.org/licenses/LICENSE-2.0  
*
```

- \* Unless required by applicable law or agreed to in writing, software
- \* distributed under the License is distributed on an "AS IS" BASIS,
- \* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
- \* See the License for the specific language governing permissions and
- \* limitations under the License.

## **W3C® SOFTWARE NOTICE AND LICENSE**

---

- \* Copyright © 1994-2000 World Wide Web Consortium,
- \* (Massachusetts Institute of Technology, Institut National de
- \* Recherche en Informatique et en Automatique, Keio University).
- \* All Rights Reserved. <http://www.w3.org/Consortium/Legal/>
- \*
- \* This W3C work (including software, documents, or other related items) is
- \* being provided by the copyright holders under the following license. By
- \* obtaining, using and/or copying this work, you (the licensee) agree that
- \* you have read, understood, and will comply with the following terms and
- \* conditions:
- \*
- \* Permission to use, copy, modify, and distribute this software and its
- \* documentation, with or without modification, for any purpose and without
- \* fee or royalty is hereby granted, provided that you include the following
- \* on ALL copies of the software and documentation or portions thereof,
- \* including modifications, that you make:
- \*
- \* 1. The full text of this NOTICE in a location viewable to users of the
- \* redistributed or derivative work.
- \*
- \* 2. Any pre-existing intellectual property disclaimers, notices, or terms

\* and conditions. If none exist, a short notice of the following form  
\* (hypertext is preferred, text is permitted) should be used within the  
\* body of any redistributed or derivative code: "Copyright ©  
\* [\$date-of-software] World Wide Web Consortium, (Massachusetts  
\* Institute of Technology, Institut National de Recherche en  
\* Informatique et en Automatique, Keio University). All Rights  
\* Reserved. <http://www.w3.org/Consortium/Legal/>"  
\*  
\* 3. Notice of any changes or modifications to the W3C files, including the  
\* date changes were made. (We recommend you provide URIs to the location  
\* from which the code is derived.)  
\*  
\* THIS SOFTWARE AND DOCUMENTATION IS PROVIDED "AS IS," AND COPYRIGHT HOLDERS  
\* MAKE NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT  
\* NOT LIMITED TO, WARRANTIES OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR  
\* PURPOSE OR THAT THE USE OF THE SOFTWARE OR DOCUMENTATION WILL NOT INFRINGE  
\* ANY THIRD PARTY PATENTS, COPYRIGHTS, TRADEMARKS OR OTHER RIGHTS.  
\*  
\* COPYRIGHT HOLDERS WILL NOT BE LIABLE FOR ANY DIRECT, INDIRECT, SPECIAL OR  
\* CONSEQUENTIAL DAMAGES ARISING OUT OF ANY USE OF THE SOFTWARE OR  
\* DOCUMENTATION.  
\*  
\* The name and trademarks of copyright holders may NOT be used in advertising  
\* or publicity pertaining to the software without specific, written prior  
\* permission. Title to copyright in this software and any associated  
\* documentation will at all times remain with copyright holders.  
\*

## ZLIB LICENSE

---

\* zlib.h -- interface of the 'zlib' general purpose compression library  
version 1.2.3, July 18th, 2005

Copyright (C) 1995-2005 Jean-loup Gailly and Mark Adler

This software is provided 'as-is', without any express or implied  
warranty. In no event will the authors be held liable for any damages  
arising from the use of this software.

Permission is granted to anyone to use this software for any purpose,  
including commercial applications, and to alter it and redistribute it  
freely, subject to the following restrictions:

1. The origin of this software must not be misrepresented; you must not  
claim that you wrote the original software. If you use this software  
in a product, an acknowledgment in the product documentation would be  
appreciated but is not required.
2. Altered source versions must be plainly marked as such, and must not be  
misrepresented as being the original software.
3. This notice may not be removed or altered from any source distribution.

Jean-loup Gailly [jloup@gzip.org](mailto:jloup@gzip.org)

Mark Adler [madler@alumni.caltech.edu](mailto:madler@alumni.caltech.edu)

## GENERAL BSD LICENSE

---

Copyright (c) 1998, Regents of the University of California

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

"Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

"Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

"Neither the name of the <ORGANIZATION> nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

## GENERAL MIT LICENSE

---

Copyright (c) 1998, Regents of the Massachusetts Institute of Technology

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

## UNICODE LICENSE

---

UNICODE, INC. LICENSE AGREEMENT - DATA FILES AND SOFTWARE

Unicode Data Files include all data files under the directories <http://www.unicode.org/Public/>, <http://www.unicode.org/reports/>, and <http://www.unicode.org/cldr/data/> . Unicode Software includes any source code published in the Unicode Standard or under the directories <http://www.unicode.org/Public/>, <http://www.unicode.org/reports/>, and <http://www.unicode.org/cldr/data/>.

NOTICE TO USER: Carefully read the following legal agreement. BY DOWNLOADING, INSTALLING, COPYING OR OTHERWISE USING UNICODE INC.'S DATA FILES ("DATA FILES"), AND/OR SOFTWARE ("SOFTWARE"), YOU UNEQUIVOCALLY ACCEPT, AND AGREE TO BE BOUND BY, ALL OF THE TERMS AND CONDITIONS OF THIS AGREEMENT. IF YOU DO NOT AGREE, DO NOT DOWNLOAD, INSTALL, COPY, DISTRIBUTE OR USE THE DATA FILES OR SOFTWARE.

COPYRIGHT AND PERMISSION NOTICE

Copyright © 1991-2006 Unicode, Inc. All rights reserved. Distributed under the Terms of Use in <http://www.unicode.org/copyright.html>.

Permission is hereby granted, free of charge, to any person obtaining a copy of the Unicode data files and any associated documentation (the "Data Files") or Unicode software and any associated documentation (the "Software") to deal in the Data Files or Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, and/or sell copies of the Data Files or Software, and to permit persons to whom the Data Files or Software are furnished to do so, provided that (a) the above copyright notice(s) and this permission notice appear with all copies of the Data Files or Software, (b) both the above copyright notice(s) and this permission notice appear in associated documentation, and (c) there is clear notice in each modified Data File or in the Software as well as in the documentation associated with the Data File(s) or Software that the data or software has been modified.



THE DATA FILES AND SOFTWARE ARE PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR HOLDERS INCLUDED IN THIS NOTICE BE LIABLE FOR ANY CLAIM, OR ANY SPECIAL INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THE DATA FILES OR SOFTWARE.

Except as contained in this notice, the name of a copyright holder shall not be used in advertising or otherwise to promote the sale, use or other dealings in these Data Files or Software without prior written authorization of the copyright holder.

Unicode and the Unicode logo are trademarks of Unicode, Inc., and may be registered in some jurisdictions. All other trademarks and registered trademarks mentioned herein are the property of their respective owners

## MISCELLANEOUS ATTRIBUTIONS

---

Adobe, Acrobat, and the Acrobat Logo are registered trademarks of Adobe Systems Incorporated.

FAST Instream is a trademark of Fast Search and Transfer ASA.

HP-UX is a registered trademark of Hewlett-Packard Company.

IBM, Informix, and DB2 are registered trademarks of IBM Corporation.

Jaws PDF Library is a registered trademark of Global Graphics Software Ltd.

Kofax is a registered trademark, and Ascent and Ascent Capture are trademarks of Kofax Image Products.

Linux is a registered trademark of Linus Torvalds.

Mac is a registered trademark, and Safari is a trademark of Apple Computer, Inc.

Microsoft, Windows, and Internet Explorer are registered trademarks of Microsoft Corporation.

MrSID is property of LizardTech, Inc. It is protected by U.S. Patent No. 5,710,835. Foreign Patents Pending.

Oracle is a registered trademark of Oracle Corporation.

Portions Copyright © 1994-1997 LEAD Technologies, Inc. All rights reserved.

Portions Copyright © 1990-1998 Handmade Software, Inc. All rights reserved.

Portions Copyright © 1988, 1997 Aladdin Enterprises. All rights reserved.

## Third Party Licenses

Portions Copyright © 1997 Soft Horizons. All rights reserved.

Portions Copyright © 1995-1999 LizardTech, Inc. All rights reserved.

Red Hat is a registered trademark of Red Hat, Inc.

Sun is a registered trademark, and Sun ONE, Solaris, iPlanet and Java are trademarks of Sun Microsystems, Inc.

Sybase is a registered trademark of Sybase, Inc.

UNIX is a registered trademark of The Open Group.

Verity is a registered trademark of Autonomy Corporation plc

# Glossary

# G

## GLOSSARY

This glossary defines terms that are related to dynamic server pages. Some terms include a cross-reference to more information about that term.

### **dynamic server pages**

Files that are checked into the content server and then used to generate web pages dynamically. HCSP, HCST, HCSF, and IDOC files are all considered dynamic server pages.

### **HCSF File**

HTML Content Server Form. HTML-based on-screen data entry page that users can fill out to contribute information to the content server. An HCSF page is a dynamic content template file that is checked into Content Server with an .HCSF extension, which causes it to be dynamically resolved by the content server when viewed. Each submitted form becomes a content item in the content server.

### **HCSP File**

HTML Content Server Page. A dynamic content template file that is checked into Content Server with a .HCSP extension, which causes it to be dynamically resolved by the content server when viewed. HCSP differs from HCST in that it puts Idoc Script into HTML-style comment areas (for example, `<!--$script-->`). This allows a browser to present a statically rendered HTML view without being confused by the dynamic content. This also means that the full-text search engine can index the contents of these pages.

### **HCST File**

HTML Content Server Template. A dynamic content template file that is checked into Content Server with a .HCST extension, which causes it to be dynamically resolved by the content server when viewed. HCST differs from HCSP in that it uses standard Idoc Script syntax, so the content cannot be indexed by the full-text searching engine.

**HTML Include**

Resource type that defines *includes*, which are pieces of Idoc Script and HTML markup that can be reused in more than one template or report file. Includes are defined between `<@dynamichtml name@>` and `<@end@>` tags (for example, see the `<install_dir>/shared/config/resources/std_page.htm` file). Include calls are specified by the Idoc Script tag `<$include name$>` (for example, see the `<install_dir>/shared/config/templates/checkin_new.htm` file). HTML include resources are defined using the HTM file format.

**IDOC File**

A resource file that is checked into Content Server with an *.idoc* extension, which makes it available to other dynamic server pages (HCSP, HCST, and HCSF). IDOC files contain HTML includes that are used to generate web pages dynamically.

**Idoc Script**

Oracle's proprietary server-side script language that is used to modify the functionality and look-and-feel of Content Server products. Idoc Script tags are in the format `<$script$>`.



## B

begin form section, 3-10  
 buttons, form, 3-12

## C

common code  
   forms, 4-7  
 comparison operators  
   dynamic server pages, 2-5, 2-6  
 content attribute, 3-5  
 Content Publisher, 2-11  
   dynamic server pages, 2-4  
   nested tags, 3-5  
   repeated ResultSet tags, 3-8  
 conventions  
   naming, 2-11  
 creating  
   dynamic server pages, 2-4  
   HCSF page, 4-3  
   HCSP page, 4-1  
   HCST page, 4-1  
   IDOC page, 4-1, 4-3

## D

data section, 3-3  
   structure, 3-3  
 dDocName parameter, 2-9  
 :default suffix, 3-6  
 defining  
   form fields, 4-9  
   form information, 4-8  
   hidden fields, 4-9  
 development  
   dynamic server pages, 2-10  
   HCSF pages, 2-11  
 dID parameter, 2-9  
 DOC\_INFO\_SIMPLE service, 4-8

docLoadResourceIncludes function, 2-8  
   HCSF pages, 3-2  
   parameters, 2-9  
 dynamic server pages, 2-1, 3-1  
   comparison operators, 2-5, 2-6  
   Content Publisher, 2-4  
   creating, 2-4  
   development recommendations, 2-10  
   docLoadResourceIncludes function, 2-8  
   examples, 4-1, 4-1  
   general tips, 2-10  
   Idoc Script functions, 2-8  
   Idoc Script tags, 2-5, 2-5  
   naming conventions, 2-11  
   overview, 2-1  
   page types, 2-3  
   process, 2-2  
   referencing metadata, 2-5, 2-7  
   special characters, 2-5, 2-7  
   syntax, 2-5

## E

end of form, 3-12  
 examples  
   code for HCSF pages, 4-7  
   defining a ResultSet in XML tags, 3-7  
   dynamic server pages, 4-1, 4-1  
   form fields, 4-9  
   HCSF pages, 4-3  
   HCSP page, 4-1  
   HCST page, 4-1  
   IDOC pages, 4-1, 4-3  
 ExtraRootNodes form element, 3-6, 3-6

## F

fields  
   form, 3-11  
 file extension  
   referencing, 4-8

- file information
  - retrieving, 4-8
- files
  - HCSF, 2-4
  - HCSP, 2-3
  - HCST, 2-3
  - IDOC, 2-3
- form buttons, 3-12
- form elements, 3-6
- form end, 3-12
- form fields, 3-11
  - defining, 4-9
- form properties, 3-11
- form section, 3-10
  - begin, 3-10
  - form buttons, 3-12
  - form end, 3-12
  - form fields, 3-11
  - properties, 3-11
- forms
  - common code, 4-7
  - defining form information, 4-8
  - properties, 4-8
  - submitting, 4-9
- functions
  - docLoadResourceIncludes, 2-8
  - Idoc Script, 2-8

## H

- .hcsf file, 2-4
  - syntax, 2-5
- HCSF file
  - definition, A-1
- HCSF pages, 2-10
  - common code, 4-7
  - content attribute, 3-5
  - data section, 3-3, 3-3
  - defining form fields, 4-9
  - defining form information, 4-8
  - defining hidden fields, 4-9
  - docLoadResourceIncludes function, 3-2
  - example, 4-3
  - form buttons, 3-12
  - form elements, 3-6
  - form end, 3-12
  - form fields, 3-11
  - form properties, 3-11
  - form section, 3-10
  - HTML declaration, 3-2
  - HTML includes, 3-2
  - isFormFinished attribute, 3-4
  - load section, 3-1

- meta tag, 3-2
- metadata tags, 3-5
- nested tags, 3-5
- referencing file extensions, 4-8
- referencing XML tags, 3-6
- repeated ResultSet tags, 3-8
- ResultSets, 3-7
- resultsets attribute, 3-4
- retrieving file information, 4-8
- submitting forms, 4-9
- tips, 2-11
- variables, 3-2
- .hcsp file, 2-3
  - syntax, 2-5
- HCSP file
  - definition, A-1
- HCSP pages
  - examples, 4-1
- .hcst file, 2-3
  - syntax, 2-5
- HCST file
  - definition, A-1
- HCST pages
  - examples, 4-1
- <HEAD> section, 3-2
- hidden fields
  - defining, 4-9
- HTML declaration
  - HCSF pages, 3-2
- HTML includes
  - definition, A-2
  - HCSF pages, 3-2

## I

- idcbegindata tag, 3-3
- idcenddata tag, 3-3
- .idoc file, 2-3
  - syntax, 2-5
- IDOC file
  - definition, A-2
- IDOC pages
  - examples, 4-1, 4-3
- Idoc Script
  - definition, A-2
- Idoc Script functions
  - dynamic server pages, 2-8
- Idoc Script tags
  - dynamic server pages, 2-5, 2-5
- isEditMode variable, 4-8
- isFormFinished attribute, 3-4
- isFormSubmit variable, 4-8

**L**

load section, 3-1

**M**

meta tag, 3-2  
 metadata  
   referencing, 2-5, 2-7  
   tags, 3-5

**N**

naming conventions  
   dynamic server pages, 2-11  
 nested tags, 3-5

**O**

operators  
   dynamic server pages, 2-5, 2-6

**P**

page types  
   dynamic server pages, 2-3  
 pages  
   HCSF, 2-10  
 parameters  
   docLoadResourceIncludes function, 2-9  
 properties  
   forms, 3-11, 4-8

**R**

ref  
   prefix, 2-7, 4-8  
 referencing  
   file extension, 4-8  
   XML tags, 3-6  
 referencing metadata  
   dynamic server pages, 2-5, 2-7  
 Rendition parameter, 2-9  
 repeated ResultSet tags, 3-8  
 ResultSets  
   repeated XML tags, 3-8  
   XML tags, 3-7

resultsets attribute, 3-4  
 resultsets form element, 3-6, 3-6  
 retrieving  
   file information, 4-8  
 RevisionSelectionMethod parameter, 2-9

**S**

scriptable services, 2-9  
 sections  
   data, 3-3, 3-3  
   form, 3-10  
   <HEAD>, 3-2  
   Load, 3-1  
 services  
   DOC\_INFO\_SIMPLE, 4-8  
   scriptable, 2-9  
 special characters  
   dynamic server pages, 2-5, 2-7  
 submitting  
   forms, 4-9  
 syntax  
   dynamic server pages, 2-5  
   HCSF file, 2-5  
   HCSP file, 2-5  
   HCST file, 2-5  
   IDOC file, 2-5

**T**

tags  
   Idoc Script, 2-5  
 tips  
   dynamic server pages, 2-10  
   HCSF pages, 2-11

**V**

variables  
   HCSF pages, 3-2

**X**

XML tags, 3-4, 3-5  
   referencing, 3-6  
   repeated, 3-8

