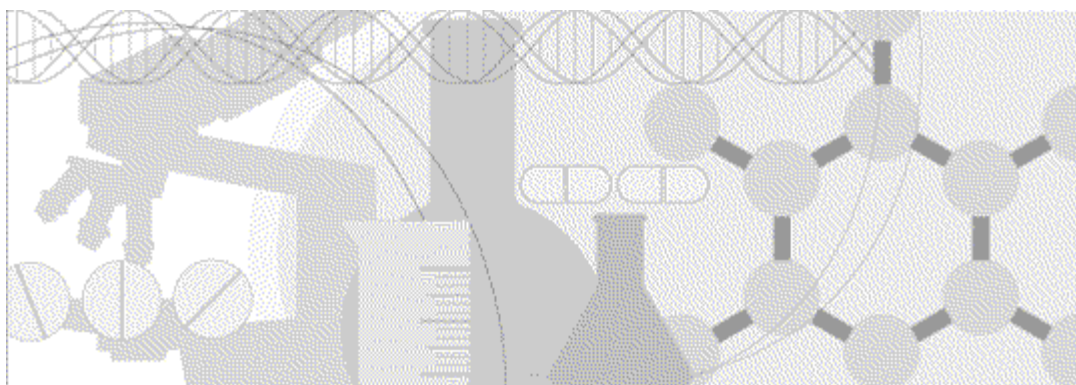


Utilities Guide

Oracle[®] Health Sciences InForm 4.6.5



ORACLE[®]

Copyright © 1998, 2015, Oracle and/or its affiliates. All rights reserved.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

This documentation may include references to materials, offerings, or products that were previously offered by Phase Forward Inc. Certain materials, offerings, services, or products may no longer be offered or provided. Oracle and its affiliates cannot be held responsible for any such references should they appear in the text provided.

Contents

About this guide	ix
Overview of this guide.....	x
Audience	x
Documentation	xi
Documentation accessibility.....	xiii
If you need assistance.....	xiv
Finding InForm information and patches on My Oracle Support.....	xiv
Finding Oracle documentation	xv
Finding prerequisite software for Oracle Health Sciences applications	xv
Chapter 1 About InForm utilities	1
Overview of InForm utilities	2
Chapter 2 PFConsole utility	3
Overview of the PFConsole utility	4
Running the PFConsole utility	5
Example	5
Using the PFConsole utility within a script.....	6
Chapter 3 MedML and the MedML Installer utility	7
Overview of MedML and the MedML Installer utility	8
What is MedML?	8
What is XML?	8
What is an XML schema?	8
What is the MedML Installer utility?.....	8
Defining a trial in the database.....	9
Defining mappings to external data formats	10
MedML Schema	11
MedML Schema Overview	11
Format for presenting MedML schema components	11
XML coding rules.....	12
Order of component definitions	14
UUIDs.....	16
Global tags.....	23
Versions.....	29
MedML Installer utility overview.....	32
Overview of the MedML Installer utility	32
Setting up a trial with the MedML Installer utility.....	32
Validation checks.....	33
Launching the MedML Installer utility	34
About the MedML Installer utility window	35
Running the MedML Installer utility.....	36
About MedML Installer utility output messages	39
Running the MedML Installer utility from the command line	39
External Data Mapping	41
About external data mapping	41
External mapping targets	41
External Data Mapping Reference.....	42
MedML Schema Reference.....	86

Action	86
Attachruledepend	87
AttachRuleSet.....	88
Browser	93
CalculatedControl.....	93
CheckboxControl	95
CodeTarget (dictionary).....	99
ContextItem (dictionary).....	100
Controlref	102
Date'imeControl	105
Dictionary (definition).....	110
DocBody.....	112
Documentation.....	113
Elementref	117
Event.....	119
Eventref.....	120
ExecutionPlan	121
ExecutionPlanref.....	122
Form	122
Formref	126
FormSet.....	127
GroupControl	134
GroupType	136
HelpLink	137
HTMLTemplate.....	138
Item	139
ItemGroup.....	144
ItemGroupref.....	145
Itemref.....	147
ItemSet	149
KeyItemref.....	152
Language	153
ManagerRef.....	154
ManagerUserGroup	155
MedMLData	156
PulldownControl	158
PfElement.....	160
QueryGroup	162
RadioControl.....	163
Report.....	170
ReportingGroup	172
Resource.....	173
Right.....	175
Rightref.....	175
RightsGroup.....	176
Rule	178
Rulearg.....	182
Section	184
Sectionref	188
Sequence.....	189
SequenceType	191
SignatureGroup.....	192
SignCRF	195
SimpleControl	196
Site.....	198
SiteGroup.....	201

Sponsor	202
StudyVersionDoc	204
StudyVersionSite.....	205
StudyVersion	206
SysConfig	209
Testcase	214
TextControl.....	216
Unit	220
Unitref	223
Update_Form_Section	224
Update_Section_Item.....	225
User.....	227
UserImage.....	230
Userref.....	232
VerbatimType	232
Scripting object reference.....	235
Conversion object	235
Data object.....	235
Execution plan objects and methods.....	235
Randomization objects and methods.....	242
Rule and calculation objects and methods.....	245

Chapter 4 InForm Data Import utility

319

Overview of the InForm Data Import utility	320
Parts of the InForm Data Import utility	320
Import methods.....	321
Special considerations.....	321
Importing a data and map file	323
Creating a data and map file	323
Specifying and editing map files.....	325
Specifying a submission type	327
Specifying an input field type	328
Building an item path.....	329
Indicating that data contains multiple selection items	331
Checking for duplicate information within itemsets	331
Importing information into an unscheduled visit.....	331
Specifying a data type.....	332
Mapping strings and child controls	332
Navigating the map file	335
Inserting or deleting an import field	335
Checking the map against the import file	336
Running the import using the data and map import file	336
Saving the map file	339
Checking the error file.....	340
Importing an XML file	341
XML import files.....	341
Creating an XML file for data import.....	341
Screening.....	344
Enrolling	344
Adding new patient clinical data	346
Updating existing patient clinical data	351
Transferring a patient record.....	356
Running the import using the XML import file	359
Importing coded items.....	363
Creating an autocode import file	363
Running the import using an autocode import file	363

Date and time validation.....	368
Additional InForm Data Import utility attributes.....	369
Running the InForm Data Import utility from the command line.....	370
Enhancing your data import.....	372
Log off of the InForm application server.....	372
Stop the WWW Publishing Service.....	372
Run the Oracle update statistics script.....	372
Change the home page.....	372
Organize by patient.....	372

Chapter 5 InForm Data Export utility 373

Overview of the InForm Data Export utility.....	374
InForm Data Export utility output options.....	374
Running the InForm Data Export utility.....	375
Exporting coded controls.....	377
Required elements.....	377
Running the export for AutoCode items.....	378
Exporting data into a CDD.....	381
Overview.....	381
Moving data to a CDD.....	381
Running the export for CDD data.....	382
Exporting Name Value Pairs.....	384
Output file format.....	385
Output file format for associated forms.....	386
Example.....	386
Exporting data in Oracle Clinical format.....	387
Overview.....	387
Transferring data to the Oracle Clinical upload application.....	387
Running the export in Oracle Clinical format.....	388
Running the InForm Data Export utility from the command line.....	392
Example.....	394
Oracle Clinical fields.....	395
Format of output files.....	396

Chapter 6 InForm Performance Monitor utility 397

Overview.....	398
Starting the InForm Performance Monitor utility.....	399
Capturing performance statistics.....	400
Viewing messages from specific subsystems.....	400
Selecting InForm servers.....	402
Performance Monitor output options.....	403
Managing the InForm Performance Monitor data.....	404
Examples of using the InForm Performance Monitor utility.....	405
Testing rule script efficiency.....	405
Reviewing SQL query performance.....	407

Chapter 7 InForm Report Folder Maintenance utility 411

Overview.....	412
Folder structure for multiple trials or sponsors.....	413
Setting up the initial folder structure.....	413
Folder structure for multiple trials.....	414
Folder structure for multiple sponsors, multiple trials.....	416
Setting up a folder structure for multiple trials or sponsors.....	418
Setting up reporting packages.....	418
Creating new folders for multiple trials or sponsors.....	419
Copying report folders.....	421

Appendix A Sample Data Import XML**423**

Overview	424
Importing screening and enrollment data	425
Importing new patient clinical data	426
Updating existing patient clinical data.....	427
Importing new itemset data	428
Editing an existing itemset	429
Deleting data from an itemset	430
Undeleting data from an itemset.....	431
Adding data to an unscheduled visit.....	432
Transferring patient records	433

About this guide

In this preface

Overview of this guide.....	x
Documentation	xi
If you need assistance.....	xiv

Overview of this guide

The *Utilities Guide* provides information about and step-by-step instructions for using the following utilities:

- PFCConsole utility
- MedML Installer utility
- InForm Data Import utility
- InForm Data Export utility
- InForm Performance Monitor utility
- InForm Report Folder Maintenance utility

Audience

This guide is for trial designers and system administrators who need to move data into and out of the InForm database.

Documentation

The product documentation is available from the following locations:

- **Oracle Software Delivery Cloud** (<https://edelivery.oracle.com>)—The complete documentation set.
- **My Oracle Support** (<https://support.oracle.com>)—*Release Notes* and *Known Issues*.
- **Oracle Technology Network** (<http://www.oracle.com/technetwork/documentation>)—The most current documentation set, excluding the *Release Notes* and *Known Issues*.

All documents may not be updated for every InForm release. Therefore, the version numbers for the documents in a release may differ.

Document	Description
<i>Release Notes</i>	The <i>Release Notes</i> document describes enhancements introduced and problems fixed in the current release, upgrade instructions, and other late-breaking information.
<i>Known Issues</i>	The <i>Known Issues</i> document provides detailed information about the known issues in this release, along with workarounds, if available.
<i>Secure Configuration Guide</i>	The <i>Secure Configuration Guide</i> provides an overview of the security features provided with the Oracle® Health Sciences InForm application, including details about the general principles of application security, and how to install, configure, and use the InForm application securely.
<i>Installation and Configuration Guide</i>	The <i>Installation and Configuration Guide</i> describes how to install the software and configure the environment for the InForm application and Cognos software.
<i>Setting Up a Trial with InForm Architect and MedML Guide</i>	The <i>Setting Up a Trial with InForm Architect and MedML Guide</i> describes how to design and implement trials in the InForm application using the InForm Architect application.
InForm Architect online Help	The InForm Architect online Help describes how to design and implement trials in the InForm application using the InForm Architect application. This information is available from the InForm Architect user interface.
<i>Step by Step for CRCs and CRAs</i>	The <i>Step by Step for CRCs and CRAs</i> describes how to use the InForm application to: <ul style="list-style-type: none"> • Screen and enroll patients. • Enter, update, and monitor clinical data. • Enter and respond to queries. • Run trial management reports and clinical data listings.
Online Help	The online Help describes how to use and administer the InForm application. This information is available from the InForm user interface.

Document	Description
<i>Reporting and Analysis Guide</i>	The <i>Reporting and Analysis Guide</i> provides an overview of the Reporting and Analysis module. It includes a brief overview of the Reporting and Analysis interface, illustrates how to access the Ad Hoc Reporting feature, and describes the trial management and clinical data packages available for Reporting and Analysis. It also provides detailed descriptions of each standard report that is included with your installation.
<i>Reporting Database Schema Guide</i>	The <i>Reporting Database Schema Guide</i> describes the Reporting and Analysis database schema.
<i>Portal Administration Guide</i>	The <i>Portal Administration Guide</i> provides step-by-step instructions for configuring and managing the InForm Portal application.
<i>Utilities Guide</i>	<p>The <i>Utilities Guide</i> provides information about and step-by-step instructions for using the following utilities:</p> <ul style="list-style-type: none"> • PFConsole utility • MedML Installer utility • InForm Data Import utility • InForm Data Export utility • InForm Performance Monitor utility • InForm Report Folder Maintenance utility
MedML Installer utility online Help	<p>The MedML Installer utility online Help provides information about, and step-by-step instructions for using, the MedML Installer utility, which is used to load XML that defines trial components into the InForm database.</p> <p>This guide also provides reference information for the MedML elements and scripting objects that are used to import and export data to and from the InForm application, as well as sample data import XML.</p> <p>This information is available from the utility user interface.</p>
InForm Data Import utility online Help	<p>The InForm Data Import utility online Help provides information about, and step-by-step instructions for using the InForm Data Import utility, which is used to import data into the InForm application.</p> <p>This information is available from the utility user interface.</p>

Document	Description
InForm Data Export utility online Help	<p>The InForm Data Export utility online Help provides information about, and step-by-step instructions for using the InForm Data Export utility, which is used to export data from the InForm application to the following output formats:</p> <ul style="list-style-type: none"> • AutoCode. • Customer-defined database (CDD). • Name value pairs. • Oracle Clinical. <p>This information is available from the utility user interface.</p>
<i>Third Party Licenses and Notices</i>	The <i>Third Party Licenses and Notices</i> document includes third party technology that may be included in or distributed with this product.

Documentation accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc>.

If you need assistance

Oracle customers have access to support through My Oracle Support. For information, visit <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info>, or if you are hearing impaired, visit <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs>.

Finding InForm information and patches on My Oracle Support

The latest information about the InForm application is on the Oracle Support self-service website, My Oracle Support. Before you install and use the InForm application, check My Oracle Support for the latest information, including *Release Notes* and *Known Issues*, alerts, white papers, bulletins, and patches.

Creating a My Oracle Support account

You must register at My Oracle Support to obtain a user name and password before you can enter the site.

- 1 Open a browser to <https://support.oracle.com>.
- 2 Click the **Register** link.
- 3 Follow the instructions on the registration page.

Finding information and articles

- 1 Sign in to My Oracle Support at <https://support.oracle.com>.
- 2 If you know the ID number of the article you need, enter the number in the text box at the top right of any page, and then click the magnifying glass icon or press **Enter**.
- 3 To search the knowledge base, click the **Knowledge** tab, and then use the options on the page to search by:
 - Product name or family.
 - Keywords or exact terms.

Finding patches

You can search for patches by patch ID or number, product, or family.

- 1 Sign in to My Oracle Support at <https://support.oracle.com>.
- 2 Click the **Patches & Updates** tab.
- 3 Enter your search criteria and click **Search**.
- 4 Click the patch ID number.

The system displays details about the patch. You can view the Read Me file before downloading the patch.

- 5 Click **Download**, and then follow the instructions on the screen to download, save, and install the patch files.

Finding Oracle documentation

The Oracle website contains links to Oracle user and reference documentation. You can view or download a single document or an entire product library.

Finding Oracle Health Sciences documentation

For Oracle Health Sciences applications, go to the Oracle Health Sciences Documentation page at <http://www.oracle.com/technetwork/documentation/hsgbu-clinical-407519.html>.

Note: Always check the Oracle Health Sciences Documentation page to ensure you have the most up-to-date documentation.

Finding other Oracle documentation

- 1 Do one of the following:
 - Go to <http://www.oracle.com/technology/documentation/index.html>.
 - Go to <http://www.oracle.com>, point to the **Support** tab, and then click **Product Documentation**.
- 2 Scroll to the product you need, and click the link.

Finding prerequisite software for Oracle Health Sciences applications

Prerequisite software for Oracle Health Sciences applications is available from the following locations:

- Download the latest major or minor release from the Oracle Software Delivery Cloud (<https://edelivery.oracle.com/>).

For information on the credentials that are required for authorized downloads, click **FAQs** on the main page of the Oracle Software Delivery Cloud portal.

- Download subsequent patch sets and patches from My Oracle Support (<https://support.oracle.com>).

To find patch sets or patches, select the **Patches & Updates** tab.

If a previous version of prerequisite software is no longer available on the Oracle Software Delivery Cloud, log a software media request Service Request (SR). Previous versions of prerequisite software are archived and can usually be downloaded. After you open an SR, you can check its status:

- US customers: Call 1-800-223-1711.
- Outside the US: Check www.oracle.com/us/support/contact/index.html for your local Oracle Support phone number.

For more information on logging a media request SR, go to My Oracle Support for Document 1071023.1: Requesting Physical Shipment or Download URL for Software Media (<https://support.oracle.com/epmos/faces/DocumentDisplay?id=1071023.1>).

CHAPTER 1

About InForm utilities

In this chapter

Overview of InForm utilities	2
------------------------------------	---

Overview of InForm utilities

Utility	Description
PFCConsole utility	Enables you to run the InForm Data Import utility, InForm Data Export utility, and MedML Installer utility from a command line.
MedML Installer utility	Installs the metadata definition of your trial into the InForm application.
InForm Data Import utility	Imports patient data into the InForm database. The following options are available: <ul style="list-style-type: none"> • MedML format. • CSV format, with which you can optionally run rules.
InForm Data Export utility	Exports patient data from the InForm database into the following formats: <ul style="list-style-type: none"> • AutoCode—Facilitates the use of external coding tools to code items. • CDD—Exports data into a Customer-Defined Database. • Name Value Pairs—Creates a pipe-delimited file that consists of data path names and data values. • Oracle® Clinical—Creates a text file that can be used to import information into an Oracle Clinical database.
InForm Performance Monitor utility	Provides statistics about several types of activities that help with performance tuning during the process of developing and implementing a trial.

CHAPTER 2

PFCConsole utility

In this chapter

Overview of the PFCConsole utility	4
Running the PFCConsole utility.....	5
Using the PFCConsole utility within a script	6

Overview of the PFConsole utility

The PFConsole utility consolidates all the activities of each tool within one window from which you can run:

- The InForm Data Import utility.
- The InForm Data Export utility.
- The MedML Installer utility.

Note: Oracle strongly recommends that you use the PFConsole utility to run these utilities from the command line. Running them individually in the command-line window is not recommended.

Running the PFConsole utility

Use the following command to open the PFConsole utility and run an InForm utility.

```
pfconsole <application> -autorun <parameters>  
where:
```

- *pfconsole* starts the PFConsole utility.
- *<application>* is one of the following:
 - *pfmminst* for the MedML Installer utility.
 - *pfimport* for the InForm Data Import utility.
 - *pfexport* for the InForm Data Export utility.
- *-autorun* is a required parameter of *pfconsole*.
- *<parameters>* are the variables for the individual utility.

For specific parameters for each utility, see:

- ***Running the MedML Installer utility from the command line*** (on page 39).
- ***Running the InForm Data Import utility from the command line*** (on page 370).
- ***Running the InForm Data Export utility from the command line*** (on page 392).

Example

```
pfconsole pfmminst -trial pfst46 -verbose -autorun -outfile text.log -xml  
filename.xml
```

Using the PFConsole utility within a script

If you are running several imports or exports, you can generate a script that will run the utilities in batch mode. To pause the script until the application completes and then moves to the next command, use the start command:

```
start /wait <pfconsole command line>
```

where:

<pfconsole command line> is the string of commands you specified when you ran the PFConsole utility.

For more information, see ***Running the PFConsole utility*** (on page 5).

CHAPTER 3

MedML and the MedML Installer utility

In this chapter

Overview of MedML and the MedML Installer utility	8
Defining a trial in the database	9
Defining mappings to external data formats	10
MedML Schema	11
MedML Installer utility overview	32
External Data Mapping.....	41
MedML Schema Reference	86
Scripting object reference	235

Overview of MedML and the MedML Installer utility

What is MedML?

MedML is a set of Extensible Markup Language (XML) tags that conform to the MedML schema (MedML.xsd), created for the purpose of defining trial components and mappings to external data formats. By using these tags, you can define the resources, forms, administrative components, and rules for a trial in a format that the MedML Installer utility can process and load into the trial database.

The MedML schema takes advantage of the structured nature of XML to enable you to reuse components throughout a trial:

- You can define a question or control once and use it on the same form multiple times or on many forms. For example, many questions on forms may require Yes or No answers. With MedML, you can create a Yes/No set of radio buttons that you can use over and over.
- You can define complex controls by creating low-level components and embedding them in higher-level components. The set of forms under a trial protocol consists of a nested set of definitions that works up from data components through data collection controls, items (questions), form sections, complete forms, and sets of forms in a visit.

The media for the InForm application includes a collection of XML files that represent predefined trial components you will be likely to use.

What is XML?

XML is a subset of Standard Generalized Markup Language (SGML), a platform-independent, international standard for the format of text and documents. XML consists of a set of tags that provide information about the format and structure of the document content they enclose. With XML, you can define your own tags and attributes to identify the structural components of a document; this is what we have done in creating the MedML schema.

What is an XML schema?

An XML schema is an XML file that defines the structure for a specific type of document. *MedML has a schema* (see "*MedML Schema*" on page 11) for loading resources, form definitions, administrative and configuration data, and rules and queries. In the files you create to load a specific type of trial component, you identify the MedML schema. To validate the structure of a MedML document, you can process it with an XML editor that checks it against the schema.

What is the MedML Installer utility?

The MedML Installer utility is a utility that parses the XML files you generate with MedML tags, validates them against the MedML schema and the constraints enforced by the database schema, and loads the trial components they define into the database. You can run the MedML Installer utility interactively by submitting a command in a DOS and GUI window, or in batch, by submitting a script containing a MedML Installer utility execution command.

Defining a trial in the database

To load a trial into the trial database:

Use the InForm Architect application to generate XML files that consist of definitions of all trial components.

Process the XML files with the MedML Installer utility. The order in which you process XML files should take into account any dependencies on data already being present in the database. For example, before you process an XML file in which you specify the sites at which a trial version is active, the sites you refer to must exist in the database. The recommended order for processing trial data XML files is:

- Resources
- Trial definition components
- Form components
- Rule components

Defining mappings to external data formats

In conjunction with defining a trial, you may want to create mappings from the trial components in an InForm trial database to external data formats. After creating mapping definitions, you can use the InForm Data Export utility to export trial data in the format you select. The MedML schema elements supports mappings to:

- Controls designed to hold data supplied by autocoding tools
- Clintrial database
- Customer-Defined Database (CDD)
- Oracle Clinical application

To load mapping definitions into the trial database:

- 1 Use the InForm Architect application to generate mapping definitions as XML files.
- 2 Process the XML files with the MedML Installer utility.

MedML Schema

MedML Schema Overview

MedML consists of the MedML schema file (MedML.xsd), which defines a standard for inserting trial implementation data into the database. The MedML schema governs the definition of:

- Resources such as images and HTML documents. Examples are GIFS used in the InForm user interface, user photographs, logos, trial documents, and online help.
- Form definition components, such as controls, forms, sets of forms (most commonly visits), and trial version.
- Administrative data such as users, groups, sites, sponsors, and configuration options. Batch insertion with an XML file that follows the MedML schema is an alternative to using the administration user interface to insert users, groups, sites, sponsors, and configuration options one by one.
- Data validation rules and their association with form definition components.
- Mapping data components between the InForm software database and a Customer-defined Database.

Format for presenting MedML schema components

This help system describes the components of the MedML schema by:

- Describing the purpose of the component.
- Listing the syntax of the component. The syntax shows the component's tag, attributes, and child components, as follows:
- `<ELEMENT
 ATTRIBUTE="value">
 <CHILD_ELEMENT/>
</ELEMENT>`
- Defining each of the component's attributes, indicating:
- Whether the attribute is required or optional
- Valid values of the attribute, including the behavior to expect if you do not submit an optional attribute
- Describing the use of any child components the parent component can have
- Presenting examples of component definitions.

The following conventions define syntax:

Convention	Meaning
REGULAR CAPITALS	Indicates an attribute name.
BOLD CAPITALS	Indicates an component name.

<i>Italic text</i>	Denotes a placeholder or variable. You must provide the actual value. For example, the attribute REFNAME=" <i>name</i> " requires you to substitute a reference name for the <i>name</i> variable.
[]	Encloses optional attributes.
	Separates an either/or choice.
+	Following a child element name, indicates that the parent component can have one or more child components.
*	Following a child component name, indicates that the parent component can have zero or more child components.
(no symbol)	Following a child component name, indicates that the parent component must have exactly one child component.

XML coding rules

When you create an XML file to load data that conforms to the MedML schema, be aware of the following coding rules.

Component definitions

Each component definition is made up of tags and attributes.

Tags

A tag consists of the component name surrounded by angle brackets (< >). For example, the tag for the *PullDownControl* (on page 158) component looks like this:

```
<PULLDOWNCONTROL>
```

Tags for components with children are paired, with an opening and closing tag. The closing tag, like the opening tag, is surrounded by angle brackets; additionally, a closing tag includes a forward slash (/). The closing tag for the PullDownControl component looks like this:

```
</PULLDOWNCONTROL>
```

When you define a compound component by specifying a component with its children, the parent component has both opening and closing tags, and the child components have an opening tag that ends with a slash. For example, in the definition of a pull-down list with four selections, which are included with <ELEMENTREF/> tags, the opening and closing tags look like this:

```
<PULLDOWNCONTROL REFNAME="MEDREASON"
  NAME="MEDREASON">
  <ELEMENTREF REFNAME="ADVERSE" ORDER="1"/>
  <ELEMENTREF REFNAME="CHRONIC" ORDER="2"/>
  <ELEMENTREF REFNAME="HORMONE" ORDER="3"/>
  <ELEMENTREF REFNAME="PROPHYL" ORDER="4"/>
</PULLDOWNCONTROL>
```

Components with no children have no separate closing tag. Instead, the component definition ends with a slash and a closing angle bracket, following any attributes that you specify. For example, the syntax for the *PFElement* (on page 160) element looks like this:

```
<PFELEMENT
  REFNAME="name"
  LABEL="name"
  [LANGUAGE="name"]
  TYPE="STRING|INTEGER|FLOAT"
  VALUE="returned_value"/>
```

Attributes

An attribute for a component uniquely identifies the component and describe how the component appears online. This online help describes the attributes for each component and indicates which are required. In component definitions, the attributes follow the component name and precede the closing angle bracket or slash and closing angle bracket. The format of an attribute is:

ATTRIBUTE_NAME = "*attribute_value*", where

- *attribute_name* is the name of the attribute as defined in the MedML schema.
- *attribute_value* is the value assigned to the attribute in the component definition. Enclose this value in double quotation marks.

This is an example of a PFELEMENT definition for a component called NA:

```
<PFELEMENT REFNAME="NA" LABEL="Not Applicable" TYPE="STRING" VALUE="NAElement" />
```

The four required attributes, REFNAME, LABEL, TYPE, and VALUE are present, and the optional attribute, LANGUAGE, is omitted. The LABEL attribute is specified in mixed case, just as the label will appear in the user interface.

Children

A child component in a component definition allows you to define controls in which multiple control definitions are nested to create a compound control. Using child components allows you to reuse previously defined components in multiple locations, by referring to them in child component definitions. For example, to define a drop-down list, you define a set of selections as components and then define the list control, including the individual selections in the list control definition as child components.

In the MedML schema, many child component names end with the letters REF (for example, *Elementref* (on page 117), *Controlref* (on page 56)) to indicate their use as references to previously defined components that are nested within the definition of another component.

When you include child components in the definition of a compound control, the child component definitions follow the opening tag of the parent component, after the parent component attributes.

The following *PullDownControl* (on page 158) definition has three child component definitions, each specifying a selection in the pull-down list.

```
<PULLDOWNCONTROL REFNAME="FRAME"
  NAME="FRAME" >
  <ELEMENTREF REFNAME="SMALL" ORDER="1" />
  <ELEMENTREF REFNAME="MEDIUM" ORDER="2" />
  <ELEMENTREF REFNAME="LARGE" ORDER="3" />
</PULLDOWNCONTROL>
```

Order of component definitions

The order in which you create component definitions is important when you load the component definitions into the database by using the MedML Installer utility. When the MedML Installer utility processes a compound component, the child components referenced in the parent component definition must be present in the database. If the child components are not in the database, the MedML Installer utility cannot create the parent component.

Therefore, when you create a set of component definitions, proceed from the bottom up, and define child components before defining the components in which they are referenced.

Additionally, the order in which you process trial definition XML files is important, as there are dependencies between the types of data that you create. For example, before you can process rules data, in which you associate a rule with a data item on a form, the item and form must exist in the database.

You can organize trial component definitions in any way that is convenient, as long as you follow the component order described in the following procedure, and you can create multiple trial component definition XML files. If you separate trial component definitions into multiple files (for example, if you create a separate file for each CRF), load them in a sequence that maintains the prescribed component order.

To create an XML file for MedML component definitions:

- 1 Create a first line that contains the xml version number. This string must be lower case:

```
<?xml version="1.0"?>
```

- 2 Add an opening and closing tag for the *MedMLData* (on page 156) component.
- 3 <MEDMLDATA> </MEDMLDATA> For version 4.0 and later of the InForm application, include the appropriate version string as the value of the xmlns attribute: <MEDMLDATA xmlns="PhaseForward-MedML- Inform4"> </MEDMLDATA>
- 4 Between the MedMLData tags, enter definitions for the following types of resources:
 - *Language* (on page 153)
 - *Browser* (on page 93)
 - .gif, .jpg, or text *Resource* (on page 173) files.
 - *HTMLTemplate* (on page 138)
 - *Report* (on page 170)
 - *Documentation* (on page 113)
 - *StudyVersionDoc* (on page 204)
- 5 Build trial data definitions in the following order:
 - *SysConfig* (on page 209)
 - *SequenceType* (on page 191)
 - *Sequence* (on page 189)
 - *Right* (on page 175)
 - *GroupType* (on page 136)

- *User* (on page 227)
 - *UserImage* (on page 230)
 - *Site* (on page 198), including references to Users who serve as site contacts, if applicable.
 - *Sponsor* (on page 202), including references to Users who serve as sponsor contacts, if applicable.
 - *QueryGroup* (on page 162), including references to Users who are members of the group.
 - *SignatureGroup* (on page 192), including references to Users who are members of the group.
 - *SiteGroup* (on page 201), including references to Users who have access to the site.
 - *ManagerUserGroup* (on page 155), including references to Users whose data will be visible in CRA reports and Users who have access to those reports.
 - *RightsGroup* (on page 176), including references to the rights that make up the group and references to Users who are assigned the rights.
 - *SignCRF* (on page 195)
- 6 Build definitions of the individual controls that appear in forms, in the following order:
 - *Unit* (see "*Right*" on page 175)
 - *PFElement* (see "*GroupType*" on page 136)
 - *SimpleControl* (see "*User*" on page 227)
 - *TextControl* (see "*Site*" on page 198)
 - *PullDownControl* (see "*Sponsor*" on page 202)
 - *DateTimeControl* (on page 105)
 - *RadioControl* (on page 163)
 - *CheckBoxControl* (on page 95)
 - *GroupControl* (on page 134)
 - *CalculatedControl* (on page 93)
 - 7 Build *Item* (on page 139) definitions by linking text questions with the controls in which a user enters data.
 - 8 Build *Itemset* (on page 149) definitions by specifying the *Items* (see "*Item*" on page 139) of which each is composed.
 - 9 Build *Section* (on page 184) definitions that include references to Items.
 - 10 Build *Form* (on page 122) definitions that consist of one or more Sections.
 - 11 Build a *StudyVersion* (on page 206) that specifies the trial protocol version and includes sets of Forms grouped as *FormSets* (see "*FormSet*" on page 127).
 - 12 Create *StudyVersionSite* (on page 205) definitions that specify which StudyVersion each site is using.

- 13 Build rule definitions from the bottom up, in the following order:
 - a **ExecutionPlan** (on page 121), if applicable.
 - b **Event** (on page 119), including a query definition and a reference to one or more ExecutionPlan definitions, if applicable.
 - c **Rule** (on page 178), including a reference to the rule event.
- 14 Use **AttachRuleSet** (on page 88) definitions to associate each rule with an item on a form.

Note: Before you can use the StudyVersionSite component to specify the trial version that a site is using, you must run the XML file that loads site definitions. Because you attach rules to form items, you must load rule definition files into the database after you load the metadata .xml files that define the form items. If you use the User attribute that enables you to specify a file with the user's picture, you must load trial definition files into the database after you load the resource .xml files that define user images. If you separate trial data definitions into multiple files, load them in a sequence that maintains the prescribed component order.

UUIDs

Universal Unique Identifiers (UUIDs) ensure the unique identification of components across all servers, databases, and trials.

In most cases, assignment of a UUID attribute is optional. However, for certain purposes, the InForm application requires the use of specific, well-known UUIDs. For example, if you want to enable users to define patient numbers, rather than allowing the InForm application to generate them automatically, you must use well-known UUIDs for the formset, form, section, item, and text control definitions that make up the specification of the patient number data entry field.

Note: The InForm Architect application and the MedML Installer utility convert alphabetic characters in UUIDs to uppercase.

If you want to change a UUID, you must install the change using the MedML Installer utility in nonstrict mode.

The following cases require a specific well-known UUID:

- *Common forms* (on page 17)
- *Date Of Visit* (on page 18)
- *Enrollment forms* (on page 18)
- *Group types* (on page 18)
- *Patient ID form* (on page 19)
- *Patient Initials item* (on page 19)
- *Patient Number item* (on page 20)
- *Randomization* (on page 20)
- *RegDocs* (on page 20)
- *Repeating visits* (on page 21)
- *Screening forms* (on page 21)
- *Sequences* (on page 21)
- *Study Completion* (on page 22)
- *Visit Report* (on page 23)

Common forms

Common forms can appear in multiple visits and have cumulative data that appears in each visit where the form is included.

To set up a common form, use the following UUID in the definition of the COMMONCRF FormSet in which the form is included:

Component	UUID
Common Form	9d6bbc5d-5811-11d2-8065- 00a0c9af7674

Once you define a common CRF by including it in a COMMONCRF FormSet, and data for any patient has been entered in it, you should not revert the form to regular CRF status by removing it from the COMMONCRF FormSet in the StudyVersion definition. Similarly, you should not change a CRF in a regular VISIT FormSet into a common CRF by adding it to a COMMONCRF FormSet after it contains data for any patient.

Note: If you attempt to change a StudyVersion in either of these ways, you will lose patient data.

Do not use a common form as the first form of a visit. When the form is reused in other visits, the Date of Visit control can't capture the specific date of each visit.

If you need to create a regular CRF that captures the same data as an existing common CRF, create it as a separate Form definition with a different REFNAME from the common CRF, and add it to the appropriate VISIT FormSets in the StudyVersion.

Date Of Visit

The Date of Visit tag appears on the first form of every visit. To set up a scheduled or unscheduled visit, you must use the following UUIDs to create a Section, Item, and DateTimeControl in which to capture the date and time of the visit:

Form component	UUID
DateTimeControl	BD991BC0-B0A4-11D2-80E3-00A0C9AF7674
Item	BD991BBF-B0A4-11D2-80E3-00A0C9AF7674
Section	BD991BBE-B0A4-11D2-80E3-00A0C9AF7674

Do not use a common form as the first form of a visit. When the form is reused in other visits, the Date of Visit control can't capture the specific date of each visit.

Enrollment forms

The definition of an Enrollment form requires several well-known UUIDs:

Component	UUID
Enrollment Formset	d882ce3a-0f42-11d2-a419- 00a0c963e0ac
Enrollment Form	d882ce3b-0f42-11d2-a419- 00a0c963e0ac
Enrollment form Section that contains the Patient Number data entry item	abcfa388-223a-11d2-a426- 00a0c963e0ac

Additionally, if the Enrollment form includes an editable Patient Number Item, that item requires the well-known UUID listed in *Patient Number item* (on page 20).

Group types

The GROUPTYPE tag defines each of the four types of administrative groups used in the InForm application: query, rights, signature, and site. The following UUIDs are required in the definitions of GroupTypes; these are included in the Base trial:

Component	UUID
Query	AC44A6E1-112E-11d2-8BED-0060082DE9D5
Rights	FA3C6201-112E-11d2-8BED-0060082DE9D5
Signature	002E58C1-112F-11d2-8BED-0060082DE9D5
Site	A4D7B9A1-112E-11d2-8BED-0060082DE9D5
CRA Report	PF_CRA_REPORT_GROUP

Patient ID form

Use the Patient ID form to enable users to change the Patient ID after a patient is fully enrolled. The Patient ID form requires the following UUIDs:

Component	UUID
Visit containing Patient ID form	03B0D5D8-7F2C-11D2-A728-00A0C977C64B
Patient ID form	06702B62-7ED6-11D2-A728-00A0C977C64B
Patient ID section	3D25EE4B-7F1B-11D2-A728-00A0C977C64B
Either or both of the following items: The Patient Number item used on the Enrollment form A Change Initials item	Patient Number item: 3D25EE4C-7F1B- 11D2-A728-00A0C977C64B Change Initials item: D959FE72-7F1C- 11D2-A728-00A0C977C64B
Either or both of the following text box controls: Patient Number text box control, if the section contains the Patient Number item used on the Enrollment form Change Initials text box control, if the section contains a Change Initials item	Patient Number control: 433DAFF6- 7F1C-11D2-A728-00A0C977C64B Change Initials control: 433DAFF7- 7F1C-11D2-A728-00A0C977C64B

Patient Initials item

The Patient Initials Item definition is required on the Screening form. Use the following UUIDs to define the Patient Initials item:

Form component	UUID
Patient Initials Item	aeb64f16-127c-11d2-a41c-00a0c963e0ac
Patient Initials TextControl	aeb64f17-127c-11d2-a41c-00a0c963e0ac
Date of Birth Item	96cae359-126c-11d2-a41c- 00a0c963e0ac
Date of Birth Control	40ace712-217c-11d2-a425- 00a0c963e0ac
Date Screened Item	96cae356-126c-11d2-a41c- 00a0c963e0ac
Date Screened Control	96cae357-126c-11d2-a41c- 00a0c963e0ac

Patient Number item

The Patient Number Item definition is an optional item on the Enrollment form. If you allow users to edit the Patient Number, use the following required UUIDs to define the Patient Number item:

Form component	UUID
Patient Number Item	3d25ee4c-7f1b-11d2-a728- 00a0c977c64b
Patient Number TextControl	433daff6-7f1c-11d2-a728-00a0c977c64b
Enrollment form Section that contains the Patient Number data entry item	abcfa388-223a-11d2-a426- 00a0c963e0ac

Randomization

When setting up the InForm application to generate randomization numbers, you must include a well-known control, item, and section on the form where the randomization number appears. In these component definitions, use the following UUIDs:

Form component	UUID
RANDOMIZATION CalculatedControl	DC2EB0BF-4F12-11d2-9319- 00A0C9769A13
DRUGKIT Item	52AF1207-4F13-11d2-9319- 00A0C9769A13
DRUGKITSECTION Section	C0482B37-4F13-11d2-9319- 00A0C9769A13

Reg Docs

The Reg Docs item is required on the Reg Docs form. The definition of Reg Docs requires the following UUID:

Component	UUID
Reg Docs Formset	PF_UUID_REGDOCS_FORMSET
Reg Docs Form	PF_UUID_REGDOCS_FORM
Reg Docs Section	PF_UUID_REGDOCS_SECTION

Repeating visits

A FormSet definition with the REPEATING attribute set to true enables you to define an unscheduled visit with its own set of forms. To set up a visit, you must use the following UUIDs to create a Section, Item, and DateTimeControl in which to capture the date and time of the visit:

Form component	UUID
DateTimeControl	BD991BC0-B0A4-11D2-80E3-00A0C9AF7674
Item	BD991BBF-B0A4-11D2-80E3-00A0C9AF7674
Section	BD991BBE-B0A4-11D2-80E3-00A0C9AF7674

Screening forms

The definition of a Screening form requires the following well-known UUIDs:

Form component	UUID
Screening Formset	d882ce38-0f42-11d2-a419- 00a0c963e0ac
Screening Form	d882ce39-0f42-11d2-a419- 00a0c963e0ac
Screening form Section that contains the Patient Initials, Date of Birth, Date Screened	96cae354-126c-11d2-a41c- 00a0c963e0ac

Sequences

The InForm application automatically generates enrollment, query, randomization, and screening numbers as a trial progresses. These numbers are generated according to a sequence established in a Sequence definition and loaded into the database in the Base trial. The required UUIDs define the enrollment, query, randomization, and screening number sequences:

Form component	UUID
Query	c
Enrollment	eb75b898-078b-11d2-a417-00a0c963e0ac
Randomization	4F4A0246-5009-11d2-931C-00A0C9769A13
Screening	f7f1b3b8-0b5c-11d2-a418-00a0c963e0ac

Study Completion

The Study Completion form records the last time the patient was seen and whether the patient completed the trial. If the patient did not complete the trial, the Study Completion form records the reason for not completing. Use the following UUIDs in the definition of a Study Completion form.

Note: The UUIDs in the Study Completion form are independent of trial versions and apply to all patients in a trial; therefore, you cannot change trial completion metadata by creating a new trial version.

Component	UUID
Visit in which the Study Completion form is included	F4699051-69E2-11D2-8FB5-00A0C977C66A
Study Completion form	7314A6A5-316E-11d2-8F9A-00A0C977C66A
Control to indicate completion status	PF_SC_COMPLETECTL
Element objects defining the values and display text for trial completion or noncompletion	Both of the following element objects must be present in the trial definition: <ul style="list-style-type: none"> PF_SC_STUDY_COMPLETE — Indicates that the patient completed the trial. PF_SC_STUDY_INCOMPLETE — Indicates that the patient did not complete the trial.
Control to indicate the reason the patient dropped out of the trial	PF_SC_REASONCTL
Text resources mapping the internal values of noncompletion reasons to the text of column headings on the standard and ad hoc patient dropout reports	PF_SC_REASONCTL_ <i>internal_reason_value</i> <i>Internal_reason_value</i> is one of the following: <ul style="list-style-type: none"> The Value property defined for an element object in a simple or pull-down control included in the control specifying noncompletion reason. The Selection Value property defined for any other type of subordinate control included in the control specifying noncompletion reason. The default value for a subordinate control for which no Selection Value property has been defined. The default value is assigned by InForm application and is in the format !pf! <i>control_DBUID_path</i>.

Visit Report

The Visit Report item is required on the Visit Report form. The definition of Visit Report requires the following UUID:

Component	UUID
Visit Report FormSet	PF_UUID_VISITREPORT_FORMSET
Visit Report Form	PF_UUID_VISITREPORT_FORM

Global tags

Except as noted, the tags described in this topic are available for use with all tags in the MedML schema.

`<!-- -->` (*Comment*) (on page 23)

HTML formatting tags (on page 24)

HTML special characters (on page 25)

Disallowed characters (on page 28)

<!-- --> (Comment)

The comment tag allows you to insert a comment in an XML file by enclosing it with angle brackets. To begin a comment, insert the `<!--` characters before the comment text; to conclude a comment, follow the comment text with the `-->` characters.

You cannot enter a comment in the middle of a tag.

Example

```
<?xml version="1.0"?>
<!DOCTYPE MEDMLDATA SYSTEM "MedML.dtd">
<MEDMLDATA>
<!-- Define PFElements for Size pulldown -->

<PFELEMENT REFNAME="SMALL" LABEL="Small" TYPE="INTEGER"VALUE="1" />
<PFELEMENT REFNAME="MEDIUM" LABEL="Medium" TYPE="INTEGER"VALUE="2" />
<PFELEMENT REFNAME="LARGE" LABEL="Large" TYPE="INTEGER"VALUE="3" />

<!-- Define SimpleControls -->

<SIMPLECONTROL REFNAME="MALE" NAME="MALE">
  <ELEMENTREF REFNAME="MALE" />
</SIMPLECONTROL>

<SIMPLECONTROL REFNAME="FEMALE" NAME="FEMALE">
  <ELEMENTREF REFNAME="FEMALE" />
</SIMPLECONTROL>
```

HTML formatting tags

The InForm application supports the following formatting tags, which you can use in any text-based trial component definitions; for example, trial protocols, CRF Help, CRF questions and notes, rule help, and FAQs:

HTML tags	Used for
	Bold text
 	Line break
<CENTER></CENTER>	Centering text an equal distance from the left and right edges of the document
<I></I>	Italic text
	List items
	Ordered (numbered) lists
<P></P>	Paragraphs
<PRE></PRE>	Preformatted plain text; for example, computer output
<S></S>	Strikethrough text
<STRIKE></STRIKE>	Strikethrough text
	Subscript text
	Superscript text
<TT></TT>	Monospace font
<U></U>	Underlined text
	Unordered (bulleted) lists

HTML special characters

The HTML specification includes numerous character sequences for specifying special characters. When you include HTML special characters in a trial component definition file, the MedML Installer utility passes the characters along to the database, and they are retrieved and processed by the forms rendering component of the InForm application.

Note: Not all special characters in the HTML specification are supported by Adobe® Portable Document Format (PDF), the output format in which the InForm Data Export utility exports printable CRFs. A complete list of special character definitions available for use in the InForm application and supported by PDF is shown below.

Character	Entity Name	Numeric Code	Descriptive
"	quotation mark	"	"
&	ampersand	&	&
<	less-than sign	<	<
>	greater-than sign	>	>
	non-breaking space	 	
¡	inverted exclamation	¡	¡
¢	cent sign	¢	¢
£	pound sterling	£	£
¤	general currency sign	¤	¤
¥	yen sign	¥	¥
§	section sign	§	§
¨	umlaut(dieresis)	¨	¨
©	copyright	©	©
^a	feminine ordinal	ª	ª
«	left angle quote, guillemotleft	«	«
¬	not sign	¬	¬
-	soft hyphen	­	­
®	registered trademark	®	®
—	macron accent	¯	¯
²	superscript two	²	²
¶	paragraph sign	¶	¶
¸	cedilla	¸	¸
º	masculine ordinal	º	º
»	right angle quote, guillemotright	»	»

¼	fraction one-fourth	¼	¼
¾	fraction three-fourths	¾	¾
¿	inverted question mark	¿	¿
Â	capital A, circumflex accent	Â	Â
Ã	capital A, tilde	Ã	Ã
Ä	capital A, dieresis or umlaut mark	Ä	Ä
Å	capital A, ring (Angstrom)	Å	Å
Æ	capital AE diphthong (ligature)	Æ	Æ
Ç	capital C, cedilla	Ç	Ç
Ê	capital E, circumflex accent	Ê	Ê
Ë	capital E, dieresis or umlaut mark	Ë	Ë
Ì	capital I, grave accent	Ì	Ì
Í	capital I, acute accent	Í	Í
Ï	capital I, dieresis or umlaut mark	Ï	Ï
Ð	capital Eth, Icelandic	Ð	Ð
Ñ	capital N, tilde	Ñ	Ñ
Ø	capital O, slash	Ø	Ø
Ù	capital U, grave accent	Ù	Ù
Ú	capital U, acute accent	Ú	Ú
Û	capital U, dieresis or umlaut mark	Ü	Ü
Ý	capital Y, acute accent	Ý	Ý
Þ	capital THORN, Icelandic	Þ	Þ
ß	small sharp s, German (sz ligature)	ß	ß
ã	small a, tilde	ã	ã
ä	small a, dieresis or umlaut mark	ä	ä
å	small a, ring	å	å
æ	small ae diphthong (ligature)	æ	æ
ç	small c, cedilla	ç	ç
ð	small eth, Icelandic	ð	ð
ñ	small n, tilde	ñ	ñ
ô	small o, circumflex accent	ô	ô
õ	small o, tilde	õ	õ
ö	small o, dieresis or umlaut mark	ö	ö
ø	small o, slash	ø	ø

ù	small u, grave accent	ù	ù
ú	small u, acute accent	ú	ú
û	small u, circumflex accent	û	û
ü	small u, dieresis or umlaut mark	ü	ü
ý	small y, acute accent	ý	ý
þ	small thorn, Icelandic	þ	þ
ÿ	small y, dieresis or umlaut mark	ÿ	ÿ

Disallowed characters

To prevent rendering and other formatting problems, do not use the following special characters:

Character	Where not to use
Double quotes	Data entry text box Form title Question text Query answer text Query text Rights group name
Single quote	Question text Query answer text Query text Site name Rights group name
Apostrophe (')	Form title Question text Rights group name
\ or \\	Anywhere
> or <	Anywhere; use HTML escape character equivalents: <ul style="list-style-type: none"> • > -- &#62; or &gt; • < -- &#60; or &lt;
Comma	Rights group name
Percent sign	Rights group name
Superscript and subscript formatting specifications	Elements that will be used in pulldown controls

Additionally, avoid copying and pasting from Microsoft Word into text boxes and query text, as Word can change characters to unicode.

Versions

Versions of trial components (on page 29)

Versions of forms (on page 30)

Versions of the trial protocol (on page 30)

Versions of documents (on page 31)

Versions of trial components

As you create trial components, you will need to update them periodically. During implementation, you may need to make corrections to trial component definitions as forms go through development and Sponsor review. During a trial, you may need to add or change trial components to reflect changes in the trial protocol or to provide more clarity in how a question is asked.

To create a new version of a trial component, submit a file containing the new component definition to the MedML Installer utility.

Note: When you revise a trial component definition, you cannot change its data type, specified in the Data Type property.

The MedML Installer utility creates a revised component definition in the trial database for most component types, and the InForm application automatically uses the most recent component definitions when rendering forms in the browser. The exceptions are the PFElement, StudyVersion, StudyVersionSite, and StudyVersionDoc components:

- **PFElement** (on page 160) definitions cannot be updated by using the MedML Installer utility. If you need to change a PFElement definition, you must delete the definition from the PF_ELEMENT table by using SQL statements, or you must create a new PFElement with a different name and use the new PFElement REFNAME everywhere you used the original PFElement.
- **StudyVersion** (on page 206) revisions provide the mechanism for updating a trial version, and you must control the version numbers manually, by using the VERSIONDESCRIPTION attribute in the StudyVersion definition.
- **StudyVersionSite** (on page 205) revisions link a StudyVersion definition to the sites at which it is used. You must control the version numbers manually, by using the VERSIONDESCRIPTION attribute in the StudyVersionSite definition.
- **StudyVersionDoc** (on page 204) revisions link a StudyVersion definition to the documents that go with the StudyVersion. You must control the version numbers manually, by using the VERSIONDESCRIPTION attribute in the StudyVersionDoc definition.

Versions of forms

When a form changes after a trial has begun, because of an approved protocol change or other Sponsor request, sites use the new version of the form for as many trial patients as makes sense. For example, if a form change involves an additional item of data to collect, and the old version of the form has already been completed for a patient, a site does not attempt to collect the new data if it depends on a time that has already passed — for example, a blood draw that must occur one hour after the patient receives a dose of the trial drug. However, if the data item is not time-sensitive, a site generally tries to collect the item, and it requires the new version of the form to do so. This means that, when you create a new version of a form, you must consider whether multiple versions of the form will be needed under the same StudyVersion:

- If a form change does not require sites to collect data from patients for whom the form has already been completed, revise the **Form** (on page 122) component, including additional or changed controls, **Items** (see "**Item**" on page 139), or **Sections** (see "**Section**" on page 184), as appropriate.
- If a form change does require sites to collect data from patients for whom the form has already been completed, revise the **Form** (on page 122) component, including additional or changed controls, **Items** (see "**Item**" on page 139), or **Sections** (see "**Section**" on page 184), as appropriate. In the Form definition, include the `Altrefname` attribute, to indicate that you are creating an alternate definition of an existing form. When InForm application runs, it presents the alternate version of the form along with the original version.

To implement a revised form, create a new StudyVersion that includes the form, and update the **StudyVersionSite** (on page 205) component for each site where the change applies.

Versions of the trial protocol

The InForm application supports revisions to a trial protocol and allows for the timing differences between when a protocol changes and when an IRB for a site approves the use of the new version. When a protocol change is approved, the InForm application begins to use the new forms, if any, associated with the protocol change when you:

- 1 Create a new StudyVersion, including the new or changed forms, by copying or revising the definition of the old **StudyVersion** (on page 206) and updating the `VERSIONDESCRIPTION` attribute in the definition.
- 2 Deploy the new StudyVersion definition at the sites by updating the `VERSIONDESCRIPTION` attribute of each **StudyVersionSite** (on page 205) component where the change applies. For a form change that does not involve a protocol amendment, update the StudyVersionSite component for each site. For a change that does involve a protocol amendment, update the StudyVersionSite component for a site only as it receives IRB approval or other authorization (for example communication from the Sponsor when a safety-driven change requires expedited implementation with approval pending).

Load the new or updated StudyVersion and StudyVersionSite components into the database by processing them with the MedML Installer utility.

Versions of documents

InForm software supports revisions to all types of documents, including:

- Trial protocol
- CRF Help
- System Help
- Sample Case Book
- Visit Calculator
- Sponsor-provided documents

To implement a new document version:

- 1 Update the files that make up the document.
- 2 If any link counts change, or if you add or remove a file from the document, update the XML file in which you specify the **Documentation** (on page 113) definition.
- 3 Load the new or updated document files into the database by using MedML Installer utility to process the **Documentation** (on page 113) definition XML file.
- 4 If you have made any changes to CRF Help, use the MedML Installer utility to reload into the database the definition of each **Form** (on page 122) to which the changed help text applies.
- 5 Update the VERSIONDESCRIPTION attribute of the **StudyVersion** (on page 206) with which each document is associated.
- 6 Update the VERSIONDESCRIPTION attribute of the **StudyVersionDoc** (on page 204) definition to match the VERSIONDEFINITION of the StudyVersion.
- 7 Deploy the new StudyVersion definition at each site where you want users to see the new document version by updating the VERSIONDESCRIPTION attribute of each **StudyVersionSite** (on page 205) component.

Use MedML Installer utility to load the updated definitions into the database in the following order: StudyVersion, StudyVersionDoc, StudyVersionSite.

MedML Installer utility overview

Overview of the MedML Installer utility

The MedML Installer utility:

- Parses the XML files that you generate with MedML tags.
- Validates the files against the following:
 - The MedML schema file (MedML.Schema.xsd) that is provided in the Oracle base installation.
 - The constraints enforced by the database schema.
- Loads the trial components that the files define into the database.

Setting up a trial with the MedML Installer utility

The trial database stores the data collected during a specific trial, including clinical data and tables that define the following information:

- CRF contents and format.
- Time and events schedule.
- Edit checks.
- User and site information.
- Sponsor personnel.
- System configuration information.
- Workflow assignments and user permissions.

After all trial components have been defined, use the MedML Installer utility to load the following definitions:

- The Base trial, which is delivered with the InForm application and contains components that are common to trials.
- The trial-specific definitions for components in your trial. These definitions are found in a set of XML files.

To specify the information that defines a trial in a trial database:

- 1 Create a set of XML files that define all of the components of a trial, using the MedML schema.
- 2 Process the XML files through the MedML Installer utility.

For more information, see the *Setting Up a Trial with InForm Architect and MedML Guide*.

Validation checks

When you install metadata with the MedML Installer utility, the program checks the metadata to make sure that the MedML syntax is correct and that all required elements are present.

Selection value checks

The MedML Installer utility performs checks that affect selection values in radio or checkbox group controls. These checks enforce selection value syntax and semantics, and help identify problems before they are incorporated into a trial. Detected errors appear in the output section in the MedML Installer window and also in the output file, if you choose to create one.

The MedML Installer utility:

- Verifies that the selection value attribute is specified for all child simple controls. If the selection value attribute is incomplete, an error occurs in both strict and nonstrict modes.

Note: In nonstrict mode, the InForm Architect application allows you to load definitions that are not fully compliant with the MedML specification for trial design purposes only. Once the trial is complete, use strict mode to ensure that the definitions you load are fully compliant with the MedML specification. For more information, see the *Setting Up a Trial with InForm Architect and MedML Guide*.

- Verifies that the selection value attribute is specified for all children of a group control. If the selection value attribute is incomplete, an error occurs in both strict and nonstrict modes.
- Checks for data type consistency among all child controls based on the following criteria:
 - If there are no simple control children, all selection values are assumed to be strings and therefore, no data type consistency is enforced.
 - If one or more children are simple controls, then all simple controls must be of the same data type (for example, string, integer, or float) and user-defined selection values for non-simple controls must be of the same data type.

If you are processing in strict mode and the data type among child controls is inconsistent, an error occurs. If you are processing in nonstrict mode, a warning message appears.

- Checks for duplicate selection values. If duplicate values exist, an error occurs in both strict and nonstrict modes.
- Checks for empty selection values for non-simple child controls. If an empty selection exists, a warning message appears if you are using verbose mode in both strict and nonstrict modes.

Coding mapping checks

When you use the Central Coding application to code patient data in an InForm trial, trial designers must develop mapping definitions to specify the:

- Items to be coded.
- Items that hold coded data.
- Relationship between those items listed above and specific code targets and context items in a coding dictionary.

When installing coding mapping definitions, the MedML Installer utility checks that the:

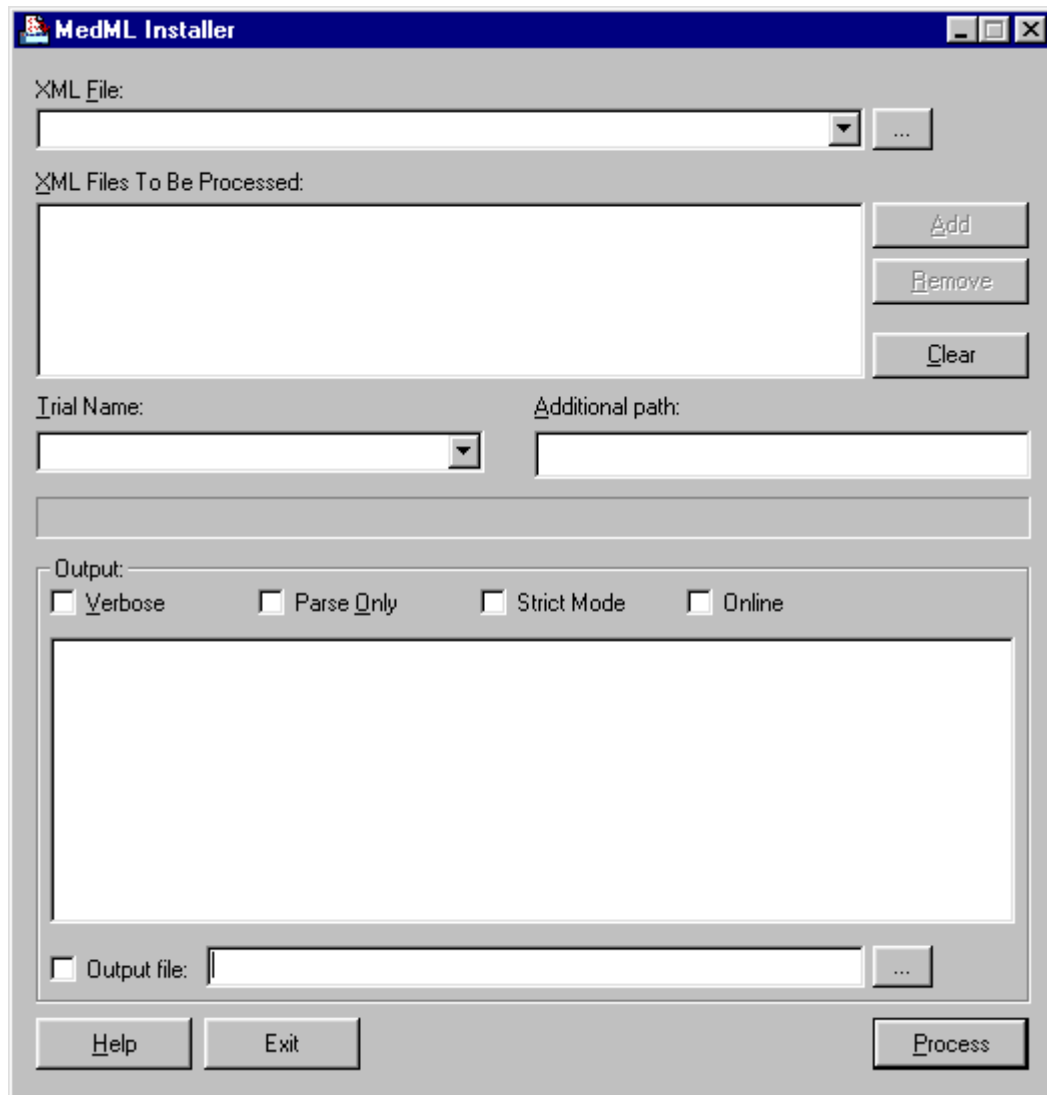
- Verbatim path exists and is complete.
- Specified dictionary is valid.
- Specified code targets and context items are valid.
- Code target and context item:
 - Names are unique within a mapping.
 - Paths exist and are complete.
- Code target paths point to top-level field controls or calculated controls.
- Repeating parts of a verbatim path and its context item paths, or a verbatim path and its code target paths, agree with each other. Verbatim and code target controls, or verbatim and context item controls, must be on the same form if the form is repeating and must be in the same itemset if the coded data appears in an itemset.
- Specified verbatim type is valid.

Launching the MedML Installer utility

To launch the MedML Installer utility, do one of the following:

- Select **Start > Programs > Oracle Health Science > InForm 4.6.5 > InForm MedML Installer**.
- Run the utility from the command line. For more information, see *Running the MedML Installer utility from the command line* (on page 39).

About the MedML Installer utility window



Field	Description
XML File	The name of the file to add to (or modify in) the build.
XML Files to be Processed	A list of all of the files to be included in the next build.
Trial Name	The name of the trial on which you are working.
Additional Path	If you are using a response (RSP) file to process a group of XML files, type the name of the root directory where the XML files that are referenced in the RSP file are located.
Verbose	Instructs the MedML Installer utility to create a detailed description of what happened during the build.
Parse Only	Instructs the MedML Installer utility to check the input file for XML errors and compatibility with the InForm database without loading data.

Field	Description
Strict Mode	<p>Instructs the MedML Installer utility to accept only complete component definitions.</p> <p>By default, the checkbox is not selected, indicating nonstrict mode. In nonstrict mode, the MedML Installer utility accepts incomplete component definitions in which not all dependent components are present. For example, it accepts a radio control definition in which not all of the element definitions have been loaded previously into the database.</p> <p>Note: This option is available only if you start the utility from the command line with the <code>-notstrict</code> option. Nonstrict mode is intended only for a trial development environment. Do not use it in production.</p> <p>Similarly, once a connection is defined, you can only load trial definition data with strict MedML processing.</p>
Online	<p>Instructs the MedML Installer utility to start the trial in online mode. By default, the checkbox is selected. In online mode, if a trial is not started when you start the MedML Installer utility, the utility starts the trial and does not shut it down when the installation is complete. This mode enables users to view the results of installing trial definition data immediately.</p> <p>If you deselect this checkbox, the MedML Installer utility starts the trial in offline mode. In offline mode, the MedML Installer utility stops the trial when installation is complete. Users cannot connect to the trial until it is restarted.</p> <p>Note: This option is available only if you start the tool from the command line with the <code>-online</code> option.</p>
Output file	A report of what happened during the build.

Definitions are processed from the bottom up. Make sure that your component definitions and XML files are organized in the correct order. The elemental components, such as controls, must be defined, imported, and read first, followed by items that reference them.

For more information, see *Running the MedML Installer utility from the command line* (on page 39).

Running the MedML Installer utility

Preparing to run the MedML Installer utility

Before you can run the MedML Installer utility, you need to know the following connection information for the server storing the trial database:

- **Trial name**—the name given to the trial database

Running the MedML Installer utility for the first time

To start testing your XML files in the trial, use the MedML Installer utility to build the files into the trial database.

To run the MedML Installer utility:

- 1 Launch the MedML Installer utility. For more information, see *Launching the MedML Installer utility* (on page 34).

The MedML Installer window appears.

- 2 In the **XML File** field, select the XML file to process, or click **Browse** and locate the file.

Note: You can also list all the files to process in a response (RSP) file and type the response file name in the XML File field.

- 3 Click **Add**.
- 4 Repeat steps 1-3 for all your XML files.
- 5 From the **Trial Name** drop-down list, select the name of the trial.
- 6 Optionally, select additional checkboxes in the MedML Installer window to specify the way that the MedML Installer utility will process the files. For more information, see *About the MedML Installer window* (on page 35).
- 7 Click **Process**.

The MedML Installer utility processes the XML files and loads information into the trial database. When all the XML files have been processed, the message **Completed Successfully** appears in the MedML Installer Output window.

For more information on error messages, see *About the MedML Installer output messages* (on page 39).

Replacing an XML file in the build

Once you have processed files using the MedML Installer utility it is not necessary to re-process every file if you need to make changes in the trial. You will need to change and save the file, figure out what files are impacted by the changes and process all those files again. You will also need to reprocess any file that references the items in the file in which you have made changes. For example, if you make a change to a section, you will have to reprocess all files that contain references to that section. If you make a change to an XML file, you will need to re-process the updated file into another build before the change appears in the trial.

- 1 Open the MedML Installer utility. The MedML Installer utility window appears.
- 2 In the XML File field, select the file in which you have made changes.
- 3 Select Add. The XML file appears in the list of XML Files to be processed.
- 4 Continue with steps 5-11 as above in Running MedML Installer for the first time. The XML file is included in the build the next time you select Process.

Updating a trial that is already in progress

When you change a file in a trial that is already in progress, you must save it, determine which files are impacted by the changes, and process those files again. You must also reprocess any file that references the items in the file to which you made changes. For example, if you make a change to a section, you must reprocess all files that contain references to that section.

When you use the MedML Installer utility to update a trial that is already in progress, you must stop and restart the trial definition with the PFAAdmin utility for the changes to take effect.

To update a trial that is already in progress:

- 1 Limit access to the trial by changing the Directory Security properties of the trial's virtual directory in **Internet Information Server** through **Microsoft Windows Components**. For more information, see the *Installation and Configuration Guide*.
- 2 Launch the MedML Installer utility.
The MedML Installer window appears.
- 3 In the **XML File** field, select the file to which you have made changes.
- 4 Click **Add**.
The XML file appears in the list of XML files to be processed.
- 5 From the **Trial Name** drop-down list, select the name of the trial.
- 6 Optionally, select additional checkboxes in the MedML Installer window to specify the way that the MedML Installer utility will process the files. For more information, see *About the MedML Installer window* (on page 35).
- 7 Click **Process**.
The MedML Installer utility processes the XML files and loads information into the trial database. When all the XML files have been processed, the message **Completed Successfully** appears in the MedML Installer Output window.
For more information on error messages, see *About the MedML Installer output messages* (on page 39).
The XML file is included in the build the next time you click **Process**.
- 8 Stop and restart the trial. For more information, see the *Installation and Configuration Guide* or the InForm online Help.

Removing XML files from the build

Regulatory authorities have strict regulations against removing data from a trial. XML files that are part of a trial cannot be removed. You can instead remove all references to any unnecessary files in the other XML files, then reprocess all the files using the MedML Installer utility.

About MedML Installer utility output messages

The MedML Installer utility generates three types of output messages:

- **Informational**—Describe conditions that probably will not cause serious problems with your build.
- **Warning**—Describe events that might be detrimental and should be fixed, but do not stop the build process.
- **Failure**—Describe events that stop the build process.

If the MedML Installer utility does not complete successfully, to determine where the error occurred:

- Check the output messages.
- Make sure that you started the InForm application before you ran the MedML Installer utility.
- Make sure that the build is in the correct directory.
- Make sure that all the parameters are accurately fulfilled.
- Check for correct paths to input files.
- Make sure components are processed in the correct order and that all necessary components are present in the XML file or database.

Running the MedML Installer utility from the command line

Note: Oracle strongly recommends that you run the MedML Installer utility through the PFConsole utility.

To run the MedML Installer utility from the command line, use the following syntax:

```
PFConsole PFMMinst -trial trialname [-verbose] [-?] [-help] [-autorun] [-notstrict] [-online] [-outfile output file name] [-parse] -xml [ @ ] response_file ... [ @ ] XMLFileN
```

Note: Parameters in brackets [] are optional.

Command line parameters

Parameter	Variable	Description
PFConsole utility		Starts the PFConsole utility.
PFMMinst		Starts the MedML Installer utility.
-trial	<i>trialname</i>	The name of the trial you are building.
-verbose		Specifies whether to generate details of the build and display them during the build.
-?		Displays the syntax of the command line for running the MedML Installer utility.
-help		Opens the online help for the MedML Installer utility.

Parameter	Variable	Description
-autorun		Runs the MedML Installer utility in a command window. Oracle recommends that you use the PFCConsole utility to run the MedML Installer utility from the command line.
-notstrict		<p>Opens the MedML Installer dialog, which has a checkbox for specifying strict mode. By default the checkbox is empty, indicating nonstrict mode. In nonstrict mode, the MedML Installer utility accepts incomplete component definitions in which not all dependent components are present; for example, it accepts a radio control definition in which not all of the element definitions have been previously loaded into the database.</p> <p>Note: Use nonstrict mode only in a trial development environment. Never load production trial definitions in nonstrict mode. Once a connection is defined, you only load trial definition data with strict MedML processing.</p>
-online		<p>Opens the MedML Installer utility dialog box with a checkbox available for specifying online mode. By default the checkbox is selected.</p> <ul style="list-style-type: none"> • Online mode—If a trial is not started when you start the MedML Installer utility, the utility starts the trial and does not shut it down when the installation is complete. This mode allows users to view the results of installing trial definition data immediately. • Offline mode—If a trial is not started when you start the MedML Installer utility, the utility starts the trial, loads the trial definition data, and stops the trial when complete. Users cannot connect to the trial until it is restarted.
-outfile	output filename	Specifies whether to save the details of the build in an output file, and indicates the path and file name in which to save it.
-parse		Instructs the utility not to commit the build to the database. If you run the MedML Installer utility in Parse mode, all information is run against, but not committed to, the database. This mode is useful for testing for errors.
-xml	XMLFile	Specifies the name of the XML file to include in the build.
@<response>		Specifies the name of a response file that consists of a list of XML files to be processed.

External Data Mapping

About external data mapping (on page 41)

External mapping targets (on page 41)

About external data mapping

The MedML tag set includes a group of tags used to define mappings from controls on forms in an InForm trial to data formats that are external to the InForm trial database, or in the case of Autocode mappings, to other controls on forms in the same InForm trial. The InForm Architect application enables you to generate mapping definitions automatically or by using mapping definition property sheets. This help system provides descriptions of the tags in the mappings that the InForm Architect application generates.

After generating the mapping definitions, you install them in the InForm trial database by using the MedML Installer utility, just as you do with trial definitions. The InForm Data Export utility and InForm Architect application use these definitions when exporting data in the selected format from the trial database.

External mapping targets

The external entities for which you can generate mappings are:

- **Autocode**—Mappings specify target calculated controls into which coded values will be inserted after trial data has been:
 - Exported to an autocoding utility with the InForm Data Export utility.
 - Coded with the external autocoding utility.
 - Re-imported into the trial with the InForm Data Import utility.
- **Central Coding**—Mappings specify information that enables codes to be applied in the Central Coding application and returned to the InForm application:
 - Coding dictionary.
 - Controls in the InForm application to be coded (verbatim).
 - Controls in the InForm application to be used as code targets and context items
- **Clintrial**—Mappings specify target items within panels in a Clintrial trial definition.
- **Customer-Defined Database (CDD)**—Mappings specify target table columns within an external Oracle database.
- **Oracle Clinical database**—Mappings specify entries in a file that can be used as input to the Oracle Clinical Batch Upload facility.

External Data Mapping Reference

AssocCDD

Purpose

The AssocCDD tag specifies the mapping in a Customer- Defined Database of an association between two forms. An association definition is specified as a *FormSet* (on page 127) with a TYPE of RELATION.

Syntax

```
<ASSOCCDD  
  REFNAME="name"  
  [DESIGNNOTE="text"]  
  [ACTIVE="true | false"]  
  TARGETTABLE="name"  
  ASSOCREFNAME="name"  
  [LABEL="name"]/>
```

Attributes

REFNAME="name"

RefName of the CDD to which you want to map the association. Required.

DESIGNNOTE="text"

Free-form text, with a maximum of 255 characters, containing any information you want to capture about the design of the component. This information is for documentation only and is not displayed. Optional.

ACTIVE="true | false"

Indicates whether the component is active: true or false. Optional; true is the default.

TARGETTABLE="name"

Name of the target CDD table in which the components of the association will be updated. An association table consists of four columns containing the RefNames of the two forms that make up the association, along with the visit in which each form occurs. Required.

ASSOCREFNAME="name"

RefName of the *FormSet* (on page 127) that defines the association. Required.

LABEL="name"

Text label for the association. The label must be 255 characters or less. Optional.

Example

The following example illustrates the mapping of an association between the AE and CM forms in the COMMONCRF FormSet.

```
<EXTERNALMAP>

  <PATH>
    <CHAPTERREF REFNAME="COMMONCRF"/>
  </PATH>

  <ASSOCDD REFNAME="CDD_WITH_ASSOC"
    TARGETTABLE="t_assoc1"
    ASSOCREFNAME="AECM_ASSOC"/>

</EXTERNALMAP>
```

Autocode

Purpose

The Autocode tag specifies a mapping to a target calculated control into which a coded value will be inserted after trial data has been:

- 1 Exported to an autocoding utility with the InForm Data Export utility.
- 2 Coded with the external autocoding utility.
- 3 Re-imported into the trial with the InForm Data Import utility.

Syntax

```
<AUTOCODE
  REFNAME="name"
  CODETARGET="path" />
```


CDD

Purpose

The CDD tag specifies the target of one mapped control in a Customer-Defined Database.

Syntax

```
<CDD
  REFNAME="name"
  [DESIGNNOTE="text"]
  [ACTIVE="true | false"]
  TARGETTABLE="name"
  TARGETCOLUMN="name"
  TARGETCOLUMNTYPE="NUMERIC | FLOAT | DATE | SPLITDATE | STRING | TEXT"
  [TARGETCOLUMNMAXLENGTH="length"]
  [TARGETKEYTYPE="PATIENT | PATIENTVISIT | PATIENTTOFORM | PATIENTTOSECTION |
  PATIENTTOITEMSET | PATIENTTOITEM | PATIENTTOCONTROL |
  PIVOTPATIENT | PIVOTVISIT | PIVOTFORM | PIVOTSECTION"
  [PIVOTCOLUMN="true | false"]
  [LABEL="string"]/>
```

Attributes

REFNAME="name"

RefName of the CDD. Required.

DESIGNNOTE="text"

Free-form text, with a maximum of 255 characters, containing any information you want to capture about the design of the component. This information is for documentation only and is not displayed. Optional.

ACTIVE="true | false"

Indicates whether the CDD is available for use: true or false. Optional; true is the default.

TARGETTABLE="name"

Name of the target customer data table in which the mapped form item will be updated. 30 characters or less; required.

TARGETCOLUMN="name"

Name of the column in the target customer data table in which the mapped form item will be added or updated. 25 characters or less; required.

TARGETCOLUMNTYPE="NUMERIC | FLOAT | DATE | SPLITDATE | STRING | TEXT"

Type of data contained in the target column. Required:

- NUMERIC
- FLOAT
- DATE — Data for a complete date and time field. Note that when you map a date and time field, the InForm application actually creates four columns internally:
- DATE—For a complete date and time
- _DT suffix—For a date and time field with only date components
- _TM suffix—For a date and time field with only time components
- _STR suffix—For an incomplete date and time field
- SPLITDATE — Data from a DateTime control, mapped to six columns, each containing one of the date or time components of the control. Each column has the generated or specified column name and the appropriate one of the following suffixes: _Day, _Mon, _Year, _Hour, _Min, or _Sec.
- STRING — Less than 255 characters.
- TEXT — Long varchar data.

Note: The TARGETCOLUMNTYPE attribute of a column that receives data from a multiple-selection control (*CheckBoxControl* (on page 95)) must be STRING.

TARGETCOLUMNMAXLENGTH="length"

Maximum length of a target column with a TARGETCOLUMNTYPE of String, in the range 1 - 255; optional.

TARGETKEYTYPE="PATIENT | PATIENTVISIT | PATIENTTOFORM | PATIENTTOSECTION | PATIENTTOITEMSET | PATIENTTOITEM | PATIENTTOCONTROL | PIVOTPATIENT | PIVOTVISIT | PIVOTFORM | PIVOTSECTION">

For the following key types, this specifies the composition of the primary key columns of the target table. Each time a component of the primary key changes from the previous primary key submitted, the InForm application inserts a new row in the target table. Primary keys consist of the following DBUIDs and indexes:

- PATIENT — PatientID, ItemsetIndex.
- PATIENTVISIT (default) — PatientID, VisitID, ItemsetIndex, and VisitIndex.
- PATIENTTOFORM — PatientID, VisitID, ItemsetIndex, VisitIndex, and FormID.
- PATIENTTOSECTION — PatientID, VisitID, ItemsetIndex, VisitIndex, FormID, and SectionID.
- PATIENTTOITEMSET — PatientID, VisitID, ItemsetIndex, VisitIndex, FormID, SectionID, and ItemsetID.

- PATIENTTOITEM — PatientID, VisitID, ItemsetIndex, VisitIndex, FormID, SectionID, ItemsetID, and ItemID.
- PATIENTTOCONTROL — PatientID, VisitID, ItemsetIndex, VisitIndex, FormID, SectionID, ItemsetID, ItemID and five ControlIDs. A target table with this key type also contains a data label that can be used for data selection.

For the following key types, the primary key columns are PatientID, VisitID, ItemsetIndex, VisitIndex, FormID, SectionID, ItemsetID, ItemID and five ControlIDs. The key type selection determines the composition of a pivot set (a group of columns in which one column is defined as the pivot column); within a pivot set, data elements mapped to non-pivot columns are repeated in each row. Target tables with these key types also contain a data label, specified as the value of the LABEL attribute, that can be used for data selection. Pivot set keys consist of the following DBUIDs and indexes:

- PIVOTPATIENT—PatientID and VisitIndex
- PIVOTVISIT—PatientID, VisitID, and VisitIndex
- PIVOTFORM—PatientID, VisitID, FormID, and VisitIndex
- PIVOTSECTION—PatientID, VisitID, FormID, SectionID, and VisitIndex

Note: A table that has mappings for controls within an itemset cannot have any of the pivot key types.

PIVOTCOLUMN="true | false"

true or false, indicating whether the column specified in the TARGETCOLUMN attribute is a pivot column. When the TARGETKEYTYPE is PIVOTPATIENT, PIVOTVISIT, PIVOTFORM, or PIVOTSECTION, data elements mapped to non-pivot columns are repeated in each row within a pivot set.

Note: The pivot column must be the first column in the table.

LABEL="string"

Text label for the data item, enabling access to a item in a target table with the PATIENTTOCONTROL or any of the PIVOT key types. The label must be 255 characters or less. Optional.

Example

The following example illustrates the use of the CDD tag to map the DESCRIBETEXT control to a column in the CDD1 Customer-Defined Database.

```
<EXTERNALMAP>

  <PATH>
    <CHAPTERREF REFNAME="PF_ALL_VISITS"/>
    <PAGEREF REFNAME="ECG"/>
    <SECTIONREF REFNAME="CHESTXRAY"/>
    <ITEMSETREF REFNAME="1"/>
    <ITEMREF REFNAME="INTERPRET2"/>
    <CONTROLREF REFNAME="INTERPRETRADIO2"/>
    <CONTROLREF REFNAME="DESCRIBETEXT"/>
  </PATH>

  <CDD REFNAME="CDD1" KEYTYPE="PATIENT" TARGETTABLE="t_ECG"
    TARGETKEYTYPE="PATIENTVISIT" TARGETCOLUMN="COMMONDAT1"
    TARGETCOLUMNTYPE="TEXT"/>

</EXTERNALMAP>
```

Chapterref

Purpose

The Chapterref tag identifies the RefName of a *FormSet* (on page 127) as the location of a mapped control in the *Path* (on page 82) tag of a mapping definition. Include one Chapterref tag in a RefName path defined by a *Path* (on page 82) tag.

Syntax

```
<CHAPTERREF
  REFNAME="name" />
```

Attributes

REFNAME="name"

RefName of the FormSet in which the mapped source control occurs. To specify that the data should be mapped to the target specified in the mapping definition from every visit in which the specified Form/Section/Itemset/Item combination occurs, enter the special RefName "PF_ALL_VISITS" as the RefName. Required.

Example

The following example illustrates the use of the PF_ALL_VISITS refname in the Chapterref tag of a Path definition.

```
<PATH>
  <CHAPTERREF REFNAME="PF_ALL_VISITS"/>
  <PAGEREF REFNAME="ECG"/>
  <SECTIONREF REFNAME="LEADECG"/>
  <ITEMSETREF REFNAME="1"/>
  <ITEMREF REFNAME="DATEASSESS"/>
  <CONTROLREF REFNAME="COMMONDATE"/>
</PATH>
```

CodeTarget (mappings)

Purpose

The CODETARGET tag specifies the mapping of a control holding coded data in an InForm trial to a specific type of code target in the dictionary used to code the data. The CODETARGET tag is a child element in the *CODINGMAP* (on page 51) structure for a verbatim specified by the *PATH* (on page 82) tag in a set of Central Coding data mappings.

Syntax

```
< CODETARGET
  NAME="name"
  PATH="path" />
```

Attributes

NAME="name" Name of the dictionary code target used to code the data in the verbatim specified by the *PATH* (on page 82) tag and its child components. Required.

PATH="path"

RefName path of the control that holds data after it is coded. Required.

Restrictions

The MedML Installer utility and the InForm application enforce the following restrictions on PATH tags defining code target control paths:

- The code target control path must be unique within a mapping.
- The code target control must be different from the verbatim and from any code target control paths within a mapping.
- The code target control path must point to the top-level text box control or calculated control.
- Verbatim, code target, and context item controls must be:
 - In the same visit if the visit is repeating
 - On the same form if the form is repeating
 - In the same itemset if the coded data appears in an itemset
 - Different from each other

Example

In the following example, the *EXTERNALMAP* (on page 67) tag identifies the mappings for one verbatim in an InForm trial, specified with the *PATH* (on page 82) tag, to be coded with the Central Coding application.

```
<EXTERNALMAP xmlns="PhaseForward-MedML-TDE">
  <PATH>
    <CHAPTERREF REFNAME="vstCORE4"/>
    <PAGEREF REFNAME="frmChem"/>
    <SECTIONREF REFNAME="sctChem"/>
    <ITEMSETREF REFNAME="mitsLabInfo"/>
    <ITEMREF REFNAME="mitmAccNo"/>
    <CONTROLREF REFNAME="mcalLAB"/>
  </PATH>
  <CODINGMAP REFNAME="MAPPINGS3" VERBATIMTYPE="MEDPROD">
    <DICTIONARY TYPE="WHODD" VERSION="05Q4" CULTURE="en-US"/>
    <CODETARGET NAME="ATC
1.CODE"          PATH="0.vstCORE4.frmChem.sctChem.mitsLabInfo.mitmLabDate.mcalL
AB"/>
    <CONTEXTINFORMATION>
      <CONTEXTITEM
NAME="Indication"    PATH="0.vstCORE4.frmChem.sctChem.mitsLabInfo.mitmLabHi.mcal
LAB"/>
      <CONTEXTITEM NAME="Route Of
Administration"    PATH="0.vstCORE4.frmChem.sctChem.mitsLabInfo.mitmLabTest.mcal
LAB"/>
    </CONTEXTINFORMATION>
  </CODINGMAP>
</EXTERNALMAP>
```

CodingMap

Purpose

The CODINGMAP tag is a container that specifies the mappings for one verbatim item to be coded.

Syntax

```
<CODINGMAP
  REFNAME="name"
  [VERBATIMTYPE="type"]>
  <DICTIONARY attributes/>
  <CODETARGET+ attributes/>
  <CONTEXTINFORMATION*/>
</CODINGMAP>
```

Attributes

REFNAME="name" RefName of the coding mapping component containing the verbatim item to be coded. Required.

VERBATIMTYPE="type"

Type of verbatim to be coded. Optional. This value corresponds to the item type in the Central Coding application:

- AE—Adverse event
- DISEASE—Disease
- LABDATA—Lab data
- MEDPROD—Medical product

AE, DISEASE, and LABDATA are valid verbatim types for the MedDRA dictionary. MEDPROD is a valid verbatim type for the WHODD dictionary.

Children

A CODINGMAP tag has the following child elements:

- One *DICTIONARY* (see "*Dictionary (mappings)*" on page 66) tag specifying the coding dictionary to use.
- At least one *CODETARGET* (see "*CodeTarget (mappings)*" on page 49) tag specifying the mapping between the InForm control holding the data after it is coded and the dictionary code target used to code the data.
- Optionally, a *CONTEXTINFORMATION* (on page 52) tag containing one or more *CONTEXTITEM* (see "*ContextItem (mappings)*" on page 53) tags. A CONTEXTITEM tag specifies the mapping between an InForm control providing additional context information and the dictionary context item used to code the data.

Example

```
<CODINGMAP REFNAME="MAPPINGS3" VERBATIMTYPE="MEDPROD">
<DICTIONARY TYPE="WHODD" VERSION="05Q4"
CULTURE="en-US" /> <CODETARGET NAME="ATC 1.CODE"
PATH="0.vstCORE4.frmChem.sctChem.mitsLabInfo.mitmLabDate.mcallAB"
/> <CONTEXTINFORMATION> <CONTEXTITEM
NAME="Indication"
PATH="0.vstCORE4.frmChem.sctChem.mitsLabInfo.mitmLabHi.mcallAB"/>
<CONTEXTITEM NAME="Route Of
Administration"
PATH="0.vstCORE4.frmChem.sctChem.mitsLabInfo.mitmLabTest.mcallAB"
/> </CONTEXTINFORMATION></CODINGMAP>
```

ContextInformation

Purpose

The `CONTEXTINFORMATION` tag is a container that encloses the mapping definitions for context items. Controls identified as context items in the InForm application provide data to context items in a coding dictionary.

Syntax

```
<CONTEXTINFORMATION>
  <CONTEXTITEM+ attributes/>
</CONTEXTINFORMATION>
```

Children

The `CONTEXTINFORMATION` tag has one or more *CONTEXTITEM* (see "*ContextItem (mappings)*" on page 53) child elements, each specifying the mapping between a control in the InForm application and a dictionary context item.

Example

```

<CODINGMAP
  REFNAME="MAPPINGS3 "
  VERBATIMTYPE="MEDPROD" >
  <DICTIONARY
    TYPE="WHODD"
    VERSION="05Q4"
    CULTURE="en-US" />
  <CODETARGET
    NAME="ATC 1.CODE"

  PATH="0.vstCORE4.frmChem.sctChem.mitsLabInfo.mitmLabDate.mcallLAB"
  />
  <CONTEXTINFORMATION>
    <CONTEXTITEM
      NAME="Indication"

  PATH="0.vstCORE4.frmChem.sctChem.mitsLabInfo.mitmLabHi.mcallLAB" />
    <CONTEXTITEM
      NAME="Route Of Administration"

  PATH="0.vstCORE4.frmChem.sctChem.mitsLabInfo.mitmLabTest.mcallLAB"
  />
  </CONTEXTINFORMATION>
</CODINGMAP>

```

ContextItem (mappings)

Purpose

The `CONTEXTITEM` tag specifies the mapping of a control holding coded data in an InForm trial to a specific type of context item in the dictionary used to code the data. The `CONTEXTITEM` tag is a child element in the *CONTEXTINFORMATION* (on page 52) structure for a verbatim specified by the *PATH* (on page 82) tag in a set of the Central Coding application data mappings.

```

<CONTEXTITEM
  NAME="name"
  PATH="path" />

```

Attributes

NAME="name" Name of the dictionary context item used to code the data in the verbatim specified by the *PATH* (on page 82) tag and its child elements. Required.

PATH="path"

RefName path of the control that holds the data for the dictionary context item. Required.

Restrictions

The MedML Installer utility and the InForm application enforce the following restrictions on PATH tags defining context item control paths:

- The context item control must be different from any code target control paths within a mapping.
- Verbatim, code target, and context item controls must be:
 - In the same visit if the visit is repeating
 - On the same form if the form is repeating
 - In the same itemset if the coded data appears in an itemset
 - Different from each other

Example

```
<EXTERNALMAP xmlns="PhaseForward-MedML-TDE">
  <PATH>
    <CHAPTERREF REFNAME="vstCORE4"/>
    <PAGEREF REFNAME="frmChem"/>
    <SECTIONREF REFNAME="sctChem"/>
    <ITEMSETREF REFNAME="mitsLabInfo"/>
    <ITEMREF REFNAME="mitmAccNo"/>
    <CONTROLREF REFNAME="mcalLAB"/>
  </PATH>
  <CODINGMAP REFNAME="MAPPINGS3" VERBATIMTYPE="MEDPROD">
    <DICTIONARY TYPE="WHODD" VERSION="05Q4" CULTURE="en-US"/>
    <CODETARGET NAME="ATC
1.CODE"          PATH="0.vstCORE4.frmChem.sctChem.mitsLabInfo.mitmLabDate.mcalL
AB"/>
    <CONTEXTINFORMATION>
      <CONTEXTITEM
NAME="Indication"          PATH="0.vstCORE4.frmChem.sctChem.mitsLabInfo.mitmLabHi.mcal
LAB"/>
      <CONTEXTITEM NAME="Route Of
Administration"          PATH="0.vstCORE4.frmChem.sctChem.mitsLabInfo.mitmLabTest.mcal
LAB"/>
    </CONTEXTINFORMATION>
  </CODINGMAP>
</EXTERNALMAP>
```

ContextPanel

Purpose

The ContextPanel tag enables you to create a definition of a Clintrial panel in a set of mappings to be exported from the InForm database to the Clintrial Integration Server. A panel definition consists of a ContextPanel tag, which is associated with CTItem tags defining the items common to all panels and one or more CTPanel tags, each associated with a single CTItem tag defining a panel item.

Syntax

```
<CONTEXT PANEL  
  REFNAME="name"  
  [ACTIVE="true | false"]  
  [DESIGNNOTE="text"]>  
  <CTITEM* attributes/>  
</CONTEXT PANEL>
```

Attributes

REFNAME="name"

RefName of the set of Clintrial mappings. Required.

ACTIVE="true | false"

Indicates whether the component is active: true or false. Optional; true is the default.

DESIGNNOTE="text"

Free-form text, with a maximum of 255 characters, containing any information you want to capture about the design of the component. This information is for documentation only and is not displayed. Optional.

Children

The definition of a ContextPanel tag can include one or more CTItem tags, each of which defines a panel item that appears in all panels.

Example

This example illustrates a ContextPanel definition included in a mapping set called MEDIKA_MAP. The ContextPanel definition includes four CItem tags, each of which defines a panel item common to all panels.

```
<EXTERNALMAP>

  <PATH />

  <CONTEXTPANEL REFNAME="MEDIKA_MAP">
    <CITEM REFNAME="PATNUM" ITEMDATATYPE="TEXT" ISREPEAT="false"
ISREQUIRED="true"
      DBFORMAT="VARCHAR2(20)" CONTEXTTYPE="1" />
    <CITEM REFNAME="VISITID" ITEMDATATYPE="TEXT" ISREPEAT="false"
ISREQUIRED="true"
      DBFORMAT="VARCHAR2(20)" CONTEXTTYPE="2" />
    <CITEM REFNAME="FORMID" ITEMDATATYPE="TEXT" ISREPEAT="false"
ISREQUIRED="true"
      DBFORMAT="VARCHAR2(20)" CONTEXTTYPE="3" />
    <CITEM REFNAME="VISITINDEX" ITEMDATATYPE="FLOAT" ISREPEAT="true"
ISREQUIRED="true"
      DBFORMAT="NUMBER(5,2)" CONTEXTTYPE="2" />
  </CONTEXTPANEL>

</EXTERNALMAP>
```

Controlref

Purpose

The Controlref tag enables you to include the definition of one type of component in the definition of another component. A Controlref appears only as the child of a component in which it is included; it is not submitted as a stand-alone component.

Types of components in which you can include Controlrefs	Types of components you can include by using Controlrefs
--	--

CheckBoxControl (on page 95)

GroupControl (on page 134)

Item (on page 139)

RadioControl (on page 163)

Path (on page 82)

CalculatedControl (on page 93)

CheckBoxControl (on page 95)

DateTimeControl (on page 105)

GroupControl (on page 134)

PullDownControl (on page 158)

RadioControl (on page 163)

SimpleControl (on page 196)

TextControl (on page 216)

Syntax

```
<CONTROLREF  
  REFNAME="name"  
  [ORDER="n"]  
  [SELECTIONVALUE="text"]/>
```

Attributes

REFNAME="name" RefName of the component that the Controlref is including in the definition of a compound control. Required.

ORDER="n"

Sequence in which each Controlref appears in the compound control definition. Optional. If you do not specify an order, the MedML Installer utility orders the components referred to by the Controlrefs in the order in which you enter them.

SELECTIONVALUE="text"

Value of the compound control included by the Controlref. Optional; if you do not specify a value, when a user selects one of the controls in the compound control, the value stored in the database is a string consisting of the characters !pf! and the DBUID path of the selected control. For example, this attribute is useful for providing a value to store for a check box or radio control component labeled Other that is associated with a TextBoxControl.

Note: When defining compound controls with Controlref or *Unitref* (on page 223) definitions, ensure that all subordinate controls return the same data type, by defining them with the same TYPE attribute.

When defining compound controls, note that the InForm application supports a maximum of five levels of nesting. Although five levels are supported, as a design practice, you should attempt to minimize the number of nested levels to help performance.

When including a Controlref definition in a *Path* (on page 82) tag, only the REFNAME attribute is valid.

Examples

This example illustrates how to create a list of radio buttons in which the first button is a pull-down list and the second is a text box. The definition of the radio button list includes the definition of each list item as a *Controlref*. This list will be used to capture a medication regimen:

- 1 Define the pull-down list as a set of *PFElements* (see "*PFElement*" on page 160), and include them in a *PullDownControl* (on page 158) definition by using *Elementrefs* (see "*Elementref*" on page 117).


```
<PFELEMENT REFNAME="QD" LABEL="QD" TYPE="STRING" VALUE="QD"/>
<PFELEMENT REFNAME="BID" LABEL="BID" TYPE="STRING" VALUE="BID"/>
<PFELEMENT REFNAME="TID" LABEL="TID" TYPE="STRING" VALUE="TID"/>
<PFELEMENT REFNAME="QID" LABEL="QID" TYPE="STRING" VALUE="QID"/>
<PULLDOWNCONTROL REFNAME="MEDREGIMEN"
  NAME="MEDREGIMEN">
  <ELEMENTREF REFNAME="QD" ORDER="1"/>
  <ELEMENTREF REFNAME="BID" ORDER="2"/>
  <ELEMENTREF REFNAME="TID" ORDER="3"/>
  <ELEMENTREF REFNAME="QID" ORDER="4"/>
</PULLDOWNCONTROL>
```
- 3 Define the text box and its caption as a *TextControl* (on page 216).


```
<TEXTCONTROL REFNAME="MEDREGTEXT"
  NAME="MEDREGTEXT"
  HEIGHT="1"
  LENGTH="20"
  MAXLENGTH="20"
  DATATYPE="STRING"
  CAPTION="Other (specify): "
  CAPTIONALIGN="TOP"/>
```
- 5 Include the PullDownControl and TextControl in a *RadioControl* (on page 163) by using Controlrefs. Note the use of the SELECTIONVALUE attribute to assign the value "OTHER" to the Other (Specify): radio button.


```
<RADIOCONTROL REFNAME="MEDREGGROUP"
  NAME="MEDREGRADIO" LAYOUT="VERTICAL">
  <CONTROLREF REFNAME="MEDREGIMEN" ORDER="1"/>
  <CONTROLREF REFNAME="MEDREGTEXT" ORDER="2"
    SELECTIONVALUE="OTHER"/>
</RADIOCONTROL>
```

This example illustrates the use of Controlref definitions in the *Path* (on page 82) tag to identify the source control used in creating a mapping of components between the InForm database and a Customer-Defined Database.

```
<EXTERNALMAP>

  <PATH>
    <CHAPTERREF REFNAME="PF_ALL_VISITS"/>
    <PAGEREF REFNAME="ECG"/>
    <SECTIONREF REFNAME="CHESTXRAY"/>
    <ITEMSETREF REFNAME="1"/>
    <ITEMREF REFNAME="INTERPRET2"/>
    <CONTROLREF REFNAME="INTERPRETRADIO2"/>
    <CONTROLREF REFNAME="DESCRIBETEXT"/>
  </PATH>

  <CDD REFNAME="CDD1" KEYTYPE="PATIENT" TARGETTABLE="t_ECG"
    TARGETKEYTYPE="PATIENTVISIT" TARGETCOLUMN="COMMONDAT1"
    TARGETCOLUMNTYPE="TEXT"/>

</EXTERNALMAP>
```

CTItem

Purpose

The CTItem tag defines the mapping of an InForm form item to a Clintrial panel item. The attributes of CTItem describe the attributes of the item in Clintrial software.

Syntax

```
<CTITEM
  REFNAME="name"
  [DESCRIPTION="text"]
  ITEMDATATYPE="TEXT|FIXED|FLOAT|DATE|DATETIME"
  [SUBSETVALUE="text"]
  [BLOCKKEYVALUE="text"]
  [PAGEKEYVALUE="text"]
  [DATEPART="n"]
  [ISDERIVED="true|false"]
  [ISREQUIRED="true|false"]
  DBFORMAT="format"
  [SASNAME="name"]
  CONTEXTTYPE="n"
  [ISREPEAT="true|false"]
  [CODELIST="text"]
  [CHECKLIST="text"]
  [RANGELOWERBOUND="n"]
  [RANGEUPPERBOUND="n"]
  [KEYORDER="n"]
  [COPYWITHPANEL="true|false"]
```

```
[LOCKSTATUS="n"/>
```

Attributes**REFNAME="name"**

RefName of the item. Required.

DESCRIPTION="text"

Description of the item. Optional

ITEMDATATYPE="TEXT | FIXED | FLOAT | DATE | DATETIME"

Data type for the value of the item: Text, Fixed, Float, Date, or DateTime. Required:

SUBSETVALUE="text"

Value the item takes if it is the subset key for subset page sections based on the panel. Optional.

BLOCKKEYVALUE="text"

Value of the Clintrial block key. If you specify this value, it overrides the visit RefName as the block key. Optional.

PAGEKEYVALUE="text"

Value of the Clintrial page key. If you specify this value, it overrides the form RefName as the page key. Optional.

DATEPART="n"

Part of a date time control to be mapped to a Clintrial item. Use this attribute if you are mapping parts of a date time control to separate Clintrial items:

- 0—Undefined
- 1—Year
- 2—Month
- 3—Day
- 4—Hour
- 5—Minute
- 6—Second

Optional. **ISDERIVED="true | false"**

Indicates whether the value of the item is determined from a derivation associated with the panel: true or false. Optional; false is the default.

ISREQUIRED="true | false"

Indicates whether the item is required: true or false. Required.

DBFORMAT="format"

Format in which Clintrial stores values for the item in the Oracle database. Required. The Oracle database formats are VARCHAR2(*n*), DATE, NUMBER(*xx*), NUMBER(*xx*,*yy*), and NUMBER(*xx*,0).*x*

SASNAME="name"

Name of the item when data is sent to SAS through the Clintrial SAS interface. Optional; if entered, the name must be eight characters or fewer and conform to SAS naming requirements.

CONTEXTTYPE="n"

Context type of the item. Required. Context items are associated with each record in a clinical data table defined by a panel of Types, 1, 2, 3, 4, or 5 and are included in the ContextPanel definition for a protocol. Required. Types are:

- 0—Not a context item.
- 1—Patient-related context item.
- 2—Visit-related context item.
- 3—Page-related context item.
- 4—Other context item.

ISREPEAT="true | false"

Indicates whether an item is one for which multiple values can be entered within a page section.

CODELIST="text"

Name of a codelist associated with the item. Optional; CODELIST and CHECKLIST are mutually exclusive. A codelist encodes entered values. Only codes or values in the codelist can be entered as values of the item.

CHECKLIST="text"

Name of a checklist associated with the item. Optional; CODELIST and CHECKLIST are mutually exclusive. A checklist is a type of codelist used to view suggested entries for a field.

RANGELOWERBOUND="n"

Minimum value that can be entered for the value of the item. Optional.

RANGEUPPERBOUND="n"

Maximum value that can be entered for the value of the item. Optional.

KEYORDER="n"

The order in which the item appears in the concatenation of key items, if the item is part of the panel's key. Optional. 0 (not a key item) is the default.

COPYWITHPANEL="true | false"

Indicates whether the item should be included with the panel if the panel is copied: true or false. Optional, true is the default.

LOCKSTATUS="n"

Indicates whether the protocol in which the item is included is locked:

- 0—item is modifiable
- 1—item is not modifiable, but it can be reset to modifiable
- 2—item is not modifiable and cannot be made modifiable.

Optional; the default is 0. **ISKEY="true | false"**

Indicates whether the item is part of the panel's key. Optional; the default is false.

Example

This example specifies the mapping of three items in the CLIN1 panel.

```
<EXTERNALMAP>
  <PATH />
  <CONTEXTPANEL REFNAME="CLIN1">
    <CTITEM REFNAME="SUBJECT" ITEMDATATYPE="TEXT"
      DBFORMAT="DBF" ISREQUIRED="true" CONTEXTTYPE="1" />
    <CTITEM REFNAME="VISITITEM"ITEMDATATYPE="TEXT"
      DBFORMAT="DBF" ISREQUIRED="true" CONTEXTTYPE="2" />
    <CTITEM REFNAME="PAGEITEM" ITEMDATATYPE="TEXT"
      DBFORMAT="DBF" ISREQUIRED="true" CONTEXTTYPE="3" />
  </CONTEXTPANEL>
</EXTERNALMAP>
```

CTPanel**Purpose**

The CTPanel tag enables you to create a definition of a Clintrial panel in a set of mappings to be exported from the InForm database to the Clintrial Integration Server. A panel definition consists of one or more CTPanel tags, each associated with a single CTItem tag defining a panel item, and a ContextPanel tag, which is associated with CTItem tags defining the items common to all panels.

Syntax

```
<CTPANEL
  REFNAME="name"
  PANELNAME="name"
  [DESCRIPTION="text"]
  PANELTYPE="n"
  [SUBSETITEM="name"]
  [ISPROTECTED="true | false"]
  [ISVERIFIABLE="true | false"]
  ISDETAILPANEL="true | false"
  [DETAILCTITEM="name"]
  [MASTERPANEL="name"]
```

```

    [MASTERCITITEM="name"]
    [SASNAME="name"]
    [LOCKSTATUS="n"]
    [ACTIVE="true | false"]
    [DESIGNNOTE="text"]>
    <CTITEM attributes/>
</CONTEXT PANEL>

```

Attributes

REFNAME="name"

RefName of the set of Clintrialmappings. Required.

PANELNAME="name"

Name of the Clintrial panel. Required.

DESCRIPTION="text"

Description of the Clintrial panel. Optional.

PANELTYPE="n"

Clintrial panel type; required:

- 0—Non-Patient Data; data is not related to a specific patient.
- 1—1 Record per Patient; one record can be collected only once during the clinical trial for each patient.
- 2—>1 Record per Patient; multiple records can be collected once in the clinical trial for each patient.
- 3—1 Record per Patient Visit; one record can be collected for each patient visit.
- 4—>1 Record per Patient Visit; multiple records can be collected for each patient visit.
- 5—Patient Enrollment; contains one record for each enrolled patient.

SUBSETITEM="name"

Name of the item specified as the subset key for subset page sections based on the panel. Optional. A subset page section can occur multiple times on a trial page in a Type 0, Type 2, or Type 4 panel, with each value of the subset key item representing distinct rows (subsets) of data.

ISPROTECTED="true | false"

Indicates whether access rights to the panel are limited in the Clintrial application; true or false. Optional; false is the default.

ISVERIFIABLE="true | false"

Indicates whether double-entry of data in panel items is required for verification. Optional; false is the default.

ISDETAILPANEL="true | false"

Indicates whether the CTPanel definition participates in a detail page section in a master-detail relationship: true or false. Required. A master-detail relationship is a relationship between two page sections on a trial page, in which each record in one page section (the master page section) can have one or more associated records in the other section (the detail page section). During data entry the displayed records in the detail page section are associated with the selected record in the master page section.

DETAILCTITEM="name"

Name of the item identified as the detail key item, if the CTPanel definition is part of a detail page section. Optional.

MASTERPANEL="name"

PANELNAME of the master panel with which this CTPanel definition participates in a master-detail relationship. Optional. This attribute is valid only if the value of the ISDETAILPANEL attribute is true.

MASTERCTITEM="name"

Name of the item on the associated master panel that corresponds to the detail key item specified in the DETAILCTITEM attribute. Optional. This attribute is valid only if the value of the ISDETAILPANEL attribute is true.

SASNAME="name"

Name of the panel when data is sent to SAS through the Clintrial SAS interface. Optional; if entered, the name must be eight characters or fewer and conform to SAS naming requirements.

LOCKSTATUS="n"

Indicates whether the protocol in which the panel is included is locked:

- 0—panel is modifiable
- 1—panel is not modifiable, but it can be reset to modifiable
- 2—panel is not modifiable and cannot be made modifiable.

Optional; the default is 0. **ACTIVE="true | false"**

Indicates whether the component is active: true or false. Optional; true is the default.

DESIGNNOTE="text"

Free-form text, with a maximum of 255 characters, containing any information you want to capture about the design of the component. This information is for documentation only and is not displayed. Optional.

Children

The definition of a CTPanel tag includes one CTItem tag, which defines an item in the panel.

Example

The following example illustrates the definition of the INCLUS panel in the MEDIKA_MAP mapping, along with the definition of the INCLU1 item. The Path tag maps this panel to a RefName path in an InForm trial definition.

```
<EXTERNALMAP>

  <PATH>
    <CHAPTERREF REFNAME="PF_ALL_VISITS" />
    <PAGEREF REFNAME="1" />
    <SECTIONREF REFNAME="INCLUS" />
    <ITEMREF REFNAME="INCLU1" />
    <CONTROLREF REFNAME="INCLU1_c" />
  </PATH>

  <CTPANEL REFNAME="MEDIKA_MAP" PANELNAME="INCLUS" DESCRIPTION=""
    PANELTYPE="1" ISPROTECTED="false" ISDETAILPANEL="false">
    <CTITEM REFNAME="INCLU1" DESCRIPTION="Inclusion criteria"
      ITEMDATATYPE="FIXED" ISREQUIRED="false" ISREPEAT="false"
      DBFORMAT="NUMBER(1)" CONTEXTTYPE="0" ISDERIVED="false"
      CODELIST="M_YESNO" KEYORDER="0" COPYWITHPANEL="false"
    LOCKSTATUS="0" />
  </CTPANEL>

</EXTERNALMAP>
```

Dictionary (mappings)

Purpose

The `DICTIONARY` tag for mappings identifies a coding dictionary within a *CODINGMAP* (on page 51) structure that specifies the mappings for one verbatim item to be coded.

Syntax

```
<DICTIONARY
  TYPE="text"
  VERSION="text"
  CULTURE="text"/>
```

Attributes

TYPE="text" Type of dictionary, for example, MedDRA or WHODD. Required.

VERSION="text"

Dictionary version, for example, 8.1, 05Q4. Required.

CULTURE="text"

Language and culture, for example, en-US. Required.

Note: The combination of **TYPE**, **VERSION**, and **CULTURE** values uniquely identifies a dictionary.

Example

```
<CODINGMAP      REFNAME="MAPPINGS3 "
  VERBATIMTYPE="MEDPROD" >
  <DICTIONARY
    TYPE="WHODD"
    VERSION="05Q4"
    CULTURE="en-US" />
  <CODETARGET
    NAME="ATC 1.CODE"

  PATH="0.vstCORE4.frmChem.sctChem.mitsLabInfo.mitmLabDate.mcallAB"
  />
  <CONTEXTINFORMATION>
    <CONTEXTITEM
      NAME="Indication"

  PATH="0.vstCORE4.frmChem.sctChem.mitsLabInfo.mitmLabHi.mcallAB" />
    <CONTEXTITEM
      NAME="Route Of Administration"

  PATH="0.vstCORE4.frmChem.sctChem.mitsLabInfo.mitmLabTest.mcallAB"
  />
  </CONTEXTINFORMATION>
```

```
</CODINGMAP>
```

ExternalMap

Purpose

The ExternalMap tag is a wrapper for the mappings for one control in an InForm trial. The ExternalMap tag can include:

- One **Path** (on page 82) tag, which identifies the source control in the InForm trial or signals the presence of mapping targets that have no corresponding control in the InForm trial. Examples of these are:
 - Columns to be created in a Customer-Defined Database that have no associated control on a CRF.
 - Items common to all panels in a Clintrial protocol.
 - Multiple tags that specify the target of each mapping.

Syntax

```
<EXTERNALMAP>
  <PATH/>
  <CDD* attributes/>
  <CONTEXTPANEL* attributes/>
  <CTPANEL* attributes/>
  <ORACLIN* attributes/>
  <ORACPEREF* attributes/>
  <ORAFIELD* attributes/>
  <AUTOCODE* attributes/>
  <CODINGMAP* attributes/>
</EXTERNALMAP>
```

Children

The ExternalMap tag can include the following mapping definition tags:

- **Path** (on page 82)—One instance of the Path tag is required. It must be the first tag included in the ExternalMap definition. When the Path tag includes child components, it defines a path to a source control in an InForm trial. When the Path appears with no child components, it indicates that the target definitions that follow have no corresponding control in an InForm trial.
- **CDD** (on page 45)—Each optional CDD tag defines a single target column in a Customer-Defined Database.
- **ContextPanel** (on page 55)—The optional ContextPanel tag defines a set of Clintrial protocol items in that are common to all panels. Each ExternalMap tag can have only one ContextPanel tag, and the ContextPanel tag must appear immediately after the Path tag, before any CTPanel definitions.
- **CTPanel** (on page 62)—Each optional CTPanel tag defines a single target item in a Clintrial panel to be exported to the Clintrial Integration Server.
- **OraClin** (on page 75)—The OraClin tag is optional. If present, it provides general settings for a set of mappings in Oracle Clinical Batch Upload format.

- **OraCPERef** (on page 75)—The optional OraCPERef tag enables you to define a reference to a Clinically Planned Event (CPE). A CPE corresponds to a visit in an InForm trial.
- **OraField** (on page 79)—The OraField tag enables you to define the mapping to one Oracle Clinical field.
- **Autocode** (on page 43)—Each optional Autocode tag defines a target control within the InForm trial for a coded value to be imported from an external autocoding tool.
- **CodingMap** (on page 51)—Each CodingMap tag contains the mappings for one verbatim item to be coded. The specifications identify the coding dictionary, the source control path for each applicable dictionary code target, and the source control path for each applicable dictionary context item.

Example

The following example illustrates the use of the ExternalMap tag along with an unqualified **Path** (on page 82) tag to define a set of mappings for items that are common to all panels in the CLIN1 mapping. Because the items are common, they do not correspond to a specific RefName path in the InForm trial.

```
<EXTERNALMAP>
  <PATH/>
  <CONTEXTPANEL REFNAME="CLIN1">
    <ITEM REFNAME="SUBJECT" ITEMDATATYPE="TEXT"
      ITEMDATAMAXLENGTH="15" CONTEXTTYPE="1"/>
    <ITEM REFNAME="VISITITEM" ITEMDATATYPE="TEXT"
      ITEMDATAMAXLENGTH="15" CONTEXTTYPE="2"/>
    <ITEM REFNAME="PAGEITEM" ITEMDATATYPE="TEXT"
      ITEMDATAMAXLENGTH="15" CONTEXTTYPE="3"/>
  </CONTEXTPANEL>
</EXTERNALMAP>
```

In the following example the ExternalMap tag defines the mapping from a control in an InForm trial, specified by the **Path** (on page 82) tag, to a table column in a CDD, specified by the **CDD** (on page 45) tag.

```
<EXTERNALMAP>
  <PATH>
    <CHAPTERREF REFNAME="PF_ALL_VISITS"/>
    <PAGEREF REFNAME="ECG"/>
    <SECTIONREF REFNAME="CHESTXRAY"/>
    <ITEMSETREF REFNAME="1"/>
    <ITEMREF REFNAME="INTERPRET2"/>
    <CONTROLREF REFNAME="INTERPRETRADIO2"/>
    <CONTROLREF REFNAME="DESCRIBETEXT"/>
  </PATH>
  <CDD REFNAME="CDD1" KEYTYPE="PATIENT" TARGETTABLE="t_ECG"
    TARGETKEYTYPE="PATIENTVISIT" TARGETCOLUMN="COMMONDAT1"
    TARGETCOLUMNTYPE="TEXT"/>
</EXTERNALMAP>
```

In the following example the ExternalMap tag defines the mappings from a control in an InForm trial, specified by the **Path** (on page 82) tag, to multiple targets: items in two different Clintrial panels,

a CDD table column, and two fields in an Oracle Clinical Batch Upload input file.

```

<EXTERNALMAP>
  <PATH>
    <CHAPTERREF REFNAME="PF_ALL_VISITS"/>
    <PAGEREF REFNAME="ECG"/>
    <SECTIONREF REFNAME="LEADECG"/>
    <ITEMSETREF REFNAME="1"/>
    <ITEMREF REFNAME="DATEASSESS"/>
    <CONTROLREF REFNAME="COMMONDATE"/>
  </PATH>

  <!-- one clinform mapping (CLIN1) -->
  <CTPANEL REFNAME="CLIN1" PANELNAME="PANEL_ECG" PANELTYPE="2"
    SUBSETITEM="S1" ISDETAILPANEL="true" DETAILCTITEM="DC1"
    MASTERPANEL="MP1">
    <CTITEM REFNAME="DATEACCESS" ITEMDATATYPE="DATETIME"
      CONTEXTTYPE="2"/>
  </CTPANEL>

  <!-- another clinform mapping (CLIN2) -->
  <CTPANEL REFNAME="CLIN2" PANELNAME="PANEL_ECG" PANELTYPE="2"
    SUBSETITEM="S1" ISDETAILPANEL="true" DETAILCTITEM="DC1"
    MASTERPANEL="MP1">
    <CTITEM REFNAME="SOMEDATE" ITEMDATATYPE="DATETIME"
      CONTEXTTYPE="2"/>
  </CTPANEL>

  <!-- CDD mapping -->
  <CDD REFNAME="CDD1" KEYTYPE="PATIENT" TARGETTABLE="t_ECG"
    TARGETKEYTYPE="PATIENTVISIT" TARGETCOLUMN="COMMONDAT1"
    TARGETCOLUMNTYPE="DATE"/>

  <!-- Illustrate an ability to have complex target description -->
  <ORACLIN REFNAME="ORA1">
    <ORAFIELD CPEREFNAME="RN1" CPENAME="N1" DCINAME="DN1"
      DCMNAME="DM1" SUBSET="S1" QUESTIONGROUP="QG1"
      QUESTION="Q1" QUESTIONVAL="QV1"/>
    <ORAFIELD CPEREFNAME="RN2" CPENAME="N2" DCINAME="DN2"
      DCMNAME="DM2" SUBSET="S2" QUESTIONGROUP="QG2" QUESTION="Q2"
      QUESTIONVAL="QV2"/>
  </ORACLIN>
</EXTERNALMAP>

```

In the following example, the ExternalMap tag identifies the mappings for one verbatim in an InForm trial, specified with the *Path* (on page 82) tag, to be coded with the Central Coding application.

```
<EXTERNALMAP xmlns="PhaseForward-MedML-TDE">
  <PATH>
    <CHAPTERREF REFNAME="vstCORE4"/>
    <PAGEREF REFNAME="frmChem"/>
    <SECTIONREF REFNAME="sctChem"/>
    <ITEMSETREF REFNAME="mitsLabInfo"/>
    <ITEMREF REFNAME="mitmAccNo"/>
    <CONTROLREF REFNAME="mcalLAB"/>
  </PATH>
  <CODINGMAP REFNAME="MAPPINGS3" VERBATIMTYPE="MEDPROD">
    <DICTIONARY TYPE="WHODD" VERSION="05Q4" CULTURE="en-US"/>
    <CODETARGET NAME="ATC
1.CODE"          PATH="0.vstCORE4.frmChem.sctChem.mitsLabInfo.mitmLabDate.mcalL
AB"/>
    <CONTEXTINFORMATION>
      <CONTEXTITEM
NAME="Indication"    PATH="0.vstCORE4.frmChem.sctChem.mitsLabInfo.mitmLabHi.mcal
LAB"/>
      <CONTEXTITEM NAME="Route Of
Administration"    PATH="0.vstCORE4.frmChem.sctChem.mitsLabInfo.mitmLabTest.mcal
LAB"/>
    </CONTEXTINFORMATION>
  </CODINGMAP>
</EXTERNALMAP>
```

ExternalMapSet

Purpose

The ExternalMapSet tag identifies a set of external mappings with one RefName. ExternalMapSet is an optional tag.

Syntax

```
<EXTERNALMAPSET
  REFNAME="name"
  [DESIGNNOTE="text"]
  [ACTIVE="true | false"]
  TYPE="CDD | CLIFORM | ORACLIN | AUTOCODE | CODINGMAP"
  [AUTOGEN="true | false"]/>
```

Attributes

REFNAME="name"

RefName of the ExternalMapSet. Required.

DESIGNNOTE="text"

Free-form text, with a maximum of 255 characters, containing any information you want to capture about the design of the component. This information is for documentation only and is not displayed. Optional.

ACTIVE="true | false"

Indicates whether the component is active: true or false. Optional; true is the default.

TYPE="CDD | CLINFORM | ORACLIN | AUTOCODE | CODINGMAP"

Indicates the target entities for which mappings are generated. Required. Values are:

- CDD—Table columns within an external Oracle database.
- CLINFORM—Items within panels in a Clintrial trial definition.
- ORACLIN—Entries in a file that can be used as input to the Oracle Clinical Batch Upload facility.
- AUTOCODE—Calculated controls into which coded values will be inserted after trial data has been:
 - Exported to an autocoding utility with the InForm Data Export utility.
 - Coded with the external autocoding utility.
 - Re-imported into the trial with the InForm Data Import utility.
- CODINGMAP—Central Coding application..

AUTOGEN="true | false" Indicates whether InForm Architect application automatically generates new mappings of the specified type when you define new form components: true or false. Optional; false is the default. This attribute is valid only when the TYPE is CDD or CLINFORM.

Example

The following example illustrates the ExternalMapSet tag in the MedML that defines the MEDIKA-CLINICAL sample trial.

```
<EXTERNALMAPSET REFNAME="MEDIKA_MAP"TYPE="CLINFORM" ACTIVE="true"
  DESIGNNOTE="My Design note"/>
```

Itemref

Purpose

The Itemref tag enables you to include previously defined Items in the definition of a section of a form.

The Itemref tag enables you to:

- Include previously defined *Items* (see "*Item*" on page 139) in the definition of a *Section* (on page 184), *ItemSet* (on page 149), or *ItemGroup* (on page 144). Include one Itemref tag for each Item in the Section, ItemSet, or ItemGroup definition.
- Specify the RefName of an Item as the location of a mapped control in the *Path* (on page 82) tag of a mapping definition. Include one Itemref tag in a RefName path defined by a Path tag.

An Itemref appears only as the child of a *Section* (on page 184) or *Path* (on page 82) in which it is included; it is not submitted as a stand-alone component.

Syntax

```
<ITEMREF
  REFNAME="name"
  [KEYITEM="true | false"]
  [UNIQUEKEY="true | false"]
  [ORDER="n"]/>
```

Attributes

REFNAME="name"

RefName of the *Item* (on page 139) that the Itemref is including in the definition of a *Section* (on page 184) or *Path* (on page 82). Required.

KEYITEM="true | false"

Indicates whether the item that the Itemref is including is a Key Item in an *ItemSet* (on page 149) definition: true or false. Optional; false is the default. The values of Key Items are displayed in a pull-down list in the InForm application to enable users to navigate to a specific instance of the itemset. Only items with text, numeric, or date/time values can be key items. Items with compound controls, including *CheckBoxControls* (see "*CheckboxControl*" on page 95), *GroupControls* (see "*GroupControl*" on page 134), *RadioControls* (see "*RadioControl*" on page 163), or *PulldownControls* (see "*PulldownControl*" on page 158), cannot be key items.

Note: You can specify Key Items for repeating forms as well as for itemsets. To specify repeating form Key Items, use the *KeyItemref* (on page 152) tag. Items that are defined as Key Items in an itemset cannot also be Key Items in a repeating form.

UNIQUEKEY="true | false"

Indicates whether the Key Item must be unique within the *FormSet* (on page 127), *Form* (on page 122), and *ItemSet* (on page 149): true or false. Optional; false is the default. If a Key Item is defined as unique and two rows of an itemset are submitted in the same visit and form with the same Key Item value, the InForm application rejects the input of the second instance.

ORDER="n"

Sequence in which each Itemref appears in the *Section* (on page 184) definition. Optional. If you do not specify an order, the MedML Installer utility orders the Items referred to by the Itemrefs in the order in which you enter them.

Note: When including an Itemref definition in a *Path* (on page 82) tag, only the REFNAME attribute is valid.

Examples

This example illustrates the use of Itemrefs in the definition of a *Section* (on page 184) called Duration of Hypertension, which contains two *Items* (see "*Item*" on page 139). For a complete illustration of the steps needed to create a Section definition, see the *Section* (on page 184) topic.

```
<SECTION REFNAME="HYPERTENSION"
  TITLE="Duration of Hypertension">
  <ITEMREF REFNAME="HTYESNO" ORDER="1"/>
  <ITEMREF REFNAME="HTDURATION" ORDER="2"/>
</SECTION>
```

The following example shows the use of an Itemref definition in a *Path* (on page 82) tag. The CLIN Item is the location where the mapped TESTTEXT control occurs.

```
<PATH>
  <CHAPTERREF REFNAME="PF_ALL_VISITS"/>
  <PAGEREF REFNAME="LAB"/>
  <SECTIONREF REFNAME="LAB"/>
  <ITEMSETREF REFNAME="LAB"/>
  <ITEMREF REFNAME="CLIN"/>
  <CONTROLREF REFNAME="CCGROUP"/>
  <CONTROLREF REFNAME="TESTTEXT"/>
</PATH>
```

The following example shows how three key items designated as a unique key combination are included in an ItemSet definition with Itemref tags.

```
<ITEMSET REFNAME="itsDOSE" ITEMREQUIRED="true"
  SDVREQUIRED="true" UNIQUEKEY="true">
  <ITEMREF REFNAME="itmDOSEFromDate" ORDER="1"
    UNIQUEKEY="true" KEYITEM="true"/>
  <ITEMREF REFNAME="itmDOSEToDate" ORDER="2"
    UNIQUEKEY="true" KEYITEM="true"/>
  <ITEMREF REFNAME="itmDOSEBlisterpack" ORDER="3"
    UNIQUEKEY="true" KEYITEM="true"/>
  <ITEMREF REFNAME="itmDOSETotalCaps" ORDER="4"/>
  <ITEMREF REFNAME="itmDOSENum" ORDER="5"/>
  <ITEMREF REFNAME="itmDOSEMissed" ORDER="6"/>
  <ITEMREF REFNAME="itmDOSEReasChange" ORDER="7"/>
  <ITEMREF REFNAME="Item_InclComments" ORDER="8"/>
</ITEMSET>
```

ItemSetref

Purpose

The ItemSetref tag identifies the RefName of an ItemSet as the location of a mapped control in the *Path* (on page 82) tag of a mapping definition. Include one ItemSet tag in a RefName path defined by a *Path* (on page 82) tag.

Syntax

```
<ITEMSETREF
  REFNAME="name"/>
```

Attributes

REFNAME="name"

RefName of the ItemSet in which the mapped source control occurs. Required.

Example

The following example shows the LAB ItemSet as the location where the mapped TESTTEXT control occurs.

```
<PATH>
  <CHAPTERREF REFNAME="PF_ALL_VISITS"/>
  <PAGEREF REFNAME="LAB"/>
  <SECTIONREF REFNAME="LAB"/>
  <ITEMSETREF REFNAME="LAB"/>
  <ITEMREF REFNAME="CLIN"/>
  <CONTROLREF REFNAME="CCGROUP"/>
  <CONTROLREF REFNAME="TESTTEXT"/>
</PATH>
```

OraClin

Purpose

The OraClin tag enables you to provide general information about a set of mappings that can be input to the Oracle Clinical Batch Upload utility. Use one OraClin tag per set of mappings.

Syntax

```
<ORACLIN
  REFNAME="name"
  [DESIGNNOTE="text"]
  [ACTIVE="true|false"]
```

Attributes

REFNAME="name"

RefName of the Oracle Clinical mapping. This name corresponds to the RefName of the mapping defined in the *ExternalMapSet* (on page 70) tag. Required.

DESIGNNOTE="text"

Free-form text, with a maximum of 255 characters, containing any information you want to capture about the design of the component. This information is for documentation only and is not displayed. Optional.

ACTIVE="true|false"

Indicates whether the component is active: true or false. Optional; true is the default.

OraCPERef

Purpose

The OraCPERef tag enables you to define a reference to a Clinically Planned Event (CPE). A CPE corresponds to a visit in an InForm trial. In visits that are unscheduled and repeating, each instance of the visit corresponds to a subevent within a CPE in Oracle Clinical. Subevents are defined by their date and time attributes. Each set of Oracle Clinical Batch Upload utility mappings can have multiple OraCPERef definitions, one for each CPE. Use the OraCPERef tag within an *ExternalMap* (on page 67) definition.

Syntax

```
<ORACPEREF
  REFNAME="name"
  [DESIGNNOTE="text"]
  [ACTIVE="true|false"]
  CPEREFNAME="name"
  CPENAME="name"
  [DATEPATH="path"]
  [TIMEPATH="path"]
  [SPECIFICDATE="date"]
  [SPECIFICTIME="time"]
```

```
ITEMSETOUTPUT="REPEATSEQUENCE|SUBEVENTBYDATE|SUBEVENTBYROW"
```

```
[REPEATINGPAGEOUTPUT="REPEATSEQUENCE|SUBEVENTBYDATE|SUBEVENTBYROW"]/>
```

Attributes

REFNAME="name"

RefName of the Oracle Clinical mapping in which the OraCPERef participates. This name corresponds to the RefName of the mapping specified in the *OraClin* (on page 75) and *ExternalMapSet* (on page 70) tags. Required.

DESIGNNOTE="text"

Free-form text, with a maximum of 255 characters, containing any information you want to capture about the design of the component. This information is for documentation only and is not displayed. Optional.

ACTIVE="true | false"

Indicates whether the component is active: true or false. Optional; true is the default.

CPEREFNAME="name"

Name of the subevent associated with the date and time values provided in the DATEPATH, TIMEPATH, SPECIFICDATE, AND SPECIFICTIME attributes. Required.

CPENAME="name"

Name of the CPE. Required.

DATEPATH="path"

RefName path of the date to use for the subevent. Either DATEPATH or SPECIFICDATE is required, and the two attributes are mutually exclusive. A RefName path is a string that identifies the datetime control to use. Each RefName in the string is the RefName specified in the .xml file that contains the definition of the form component. The RefName path is in the following format: *(Patient.Visit.Form.Section.Itemset.Item[.control[.control...]])*

- *Patient*—Always 0.
- *Visit*—RefName of the visit, as specified in the .xml file that contains the visit definition.
- *Form*—RefName of the form, as specified in the .xml file that contains the form definition.
- *Section*—RefName of the section, as specified in the .xml file that contains the section definition.
- *Itemset*—RefName of the itemset, as specified in the .xml file that contains the itemset definition.
- *Item*—RefName of the item, as specified in the .xml file that contains the item definition.
- *Control*—RefName of the datetime control to use. To access a component of a group control, refer to each parent control in which the child component is nested. For example, to address a datetime control within a group control, enter the RefName of the group control followed by the RefName of the datetime control, and separate the names with periods, as follows:
GroupControlRefname.DateTimeControlRefname.

TIMEPATH="path"

RefName path of the time to use for the subevent. Optional. If you specify a TIMEPATH value, do not specify a value for SPECIFICTIME.

SPECIFICDATE="date"

Specific date to use for the subevent, in YYYYMMDD format. Either a DATEPATH or a SPECIFICDATE value is required; do not specify both.

SPECIFICTIME="time"

Specific time to use for the subevent, in HHMMSS format. Optional. If you specify a SPECIFICTIME value, do not specify a value for TIMEPATH.

ITEMSETOUTPUT="REPEATSEQUENCE | SUBEVENTBYDATE | SUBEVENTBYROW"

Method to use for handling itemsets in exports. Required:

- RepeatSequence—Handle itemsets as repeating sequences.
- SubeventByDate—Handle itemsets as subevents, with each different itemset date resulting in a new subevent.
- SubeventByRow—Handle itemsets as subevents, with each itemset row resulting in a new subevent, regardless of whether the date is different from the previous row.

Note: Exporting as subevents incremented by row is not recommended. Items that are not in an itemset and share the same CPE name with items that are in an itemset run the risk of having their subevent numbers being incremented by those items in itemsets.

REPEATINGPAGEOUTPUT="REPEATSEQUENCE | SUBEVENTBYDATE | SUBEVENTBYROW"

Method to use for handling instances of repeating forms in exports. Optional; RepeatSequence is the default:

- RepeatSequence — Handle repeating form instances as repeating sequences.
- SubeventByDate — Handle repeating form instances as subevents, with each different form instance date resulting in a new subevent.
- SubeventByRow — Handle repeating form instances as subevents, with each form instance resulting in a new subevent, regardless of whether the date is different from the previous instance.

Examples

The following examples illustrate several coding options of the OraCERef tag.

```
<EXTERNALMAP>
```

```
<PATH>
  <CHAPTERREF REFNAME="PF_ALL_VISITS" />
  <PAGEREF REFNAME="DOV" />
  <SECTIONREF REFNAME="DOV" />
  <ITEMSETREF REFNAME="DOV" />
  <CONTROLREF REFNAME="DOV" />
</PATH>
```

CPE Reference defined with a fixed date and time.

```
<ORACPEREF REFNAME="ORA1" CPEREFNAME="CPEREF1" CPENAME="CPE1"
  DATE="20020325" TIME="235959" ITEMSETOUTPUT="SUBEVENTBYDATE" />
```

CPE Reference defined with a fixed time only.

```
<ORACPEREF REFNAME="ORA1" CPEREFNAME="CPEREF2" CPENAME="CPE1"
  DATE="20001231" ITEMSETOUTPUT="SUBEVENTBYROW" />
```

CPE Reference defined with a date from an In InForm control and a fixed time.

```
<ORACPEREF REFNAME="ORA1" CPEREFNAME="CPEREF3" CPENAME="CPE1"
  DATEPATH="0.UnschVisit.DOV.DOV.0.DOV.DOV" TIME="235959"
  ITEMSETOUTPUT="REPEATSEQUENCE" />
```

CPE Reference defined with a RefName path to a date control and no time.

```
<ORACPEREF REFNAME="ORA1" CPEREFNAME="CPEREF4" CPENAME="CPE1"
  DATEPATH="0.UnschVisit.DOV.DOV.0.DOV.DOV"
  ITEMSETOUTPUT="REPEATSEQUENCE" />
```

CPE Reference defined with RefName path to a date control and time from the same InForm control

```
<ORACPEREF REFNAME="ORA1" CPEREFNAME="CPEREF5" CPENAME="CPE1"
  DATEPATH="0.UnschVisit.DOV.DOV.0.DOV.DOV"
  TIMEPATH="0.UnschVisit.DOV.DOV.0.DOV.DOV"
  ITEMSETOUTPUT="REPEATSEQUENCE" />
```

```
</EXTERNALMAP>
```

OraField

Purpose

The OraField tag enables you to define the mapping to one Oracle Clinical field. Use this tag within an *ExternalMap* (on page 67) definition that contains a *Path* (on page 82) tag with attributes that specify the source control in an InForm trial. The OraField definition references a previously defined *OraCPERef* (on page 75) definition.

Syntax

```
<ORAFIELD
  REFNAME="name"
  [DESIGNNOTE="text"]
  [ACTIVE="true | false"]
  CPEREFNAME="name"
  DCINAME="name"
  DCMNAME="name"
  SUBSET="text"
  QUESTIONGROUP="text"
  QUESTION="text"
  QUESTIONVAL="text"
  UPPERCASE="true | false"
  [DATETIMEFIELD="DATE | TIME | DATETIME"
  [REPSEQUENCE="text"]
  [OCCURRENCE="text"]
  [SUBEVENT="text"]/>
```

Attributes

REFNAME="name"

RefName of the Oracle Clinical field. Required.

DESIGNNOTE="text"

Free-form text, with a maximum of 255 characters, containing any information you want to capture about the design of the component. This information is for documentation only and is not displayed. Optional.

ACTIVE="true | false"

Indicates whether the component is active: true or false. Optional; true is the default.

CPEREFNAME="name"

Name of the subevent of the Clinically Planned Event (CPE) with which the field is associated. This name matches the CPEREFNAME attribute of the *OraCPERef* (on page 75) definition. Required.

DCINAME="name"

Data Collection Instrument (DCI) name, a customer- defined grouping of Data Collection Modules. Required.

DCMNAME="name"

Data Collection Module (DCM) name, a customer-defined grouping of data items. Required.

SUBSET="text"

Name of a customer-defined grouping by data type. Required.

QUESTIONGROUP="text"

Name of a customer-defined grouping by question type. Required.

QUESTION="text"

Customer-defined standard variable name for the question (for example, lab test performed). Required.

QUESTIONVAL="text"

Answer to the question, or test result. Required.

UPPERCASE="true | false"

Indicates whether the data is translated to uppercase: true or false. Required.

DATETIMEFIELD="DATE | TIME | DATETIME"

If the field is a datetime field, the pertinent part of the data: Date, Time, or DateTime. Optional.

REPSEQUENCE="text"

Number incremented for each result. You can also use a repeat sequence number in conjunction with the Occurrence number used to indicate dependant or related data. A common repeat sequence number indicates a relationship between values. Optional.

OCCURRENCE="text"

Number of test performed; used to associate repeating data, for example repeated labs. A change in occurrence number indicates a retest value (for example, initial value=0, retest value =1). Optional; default value is zero.

SUBEVENT="text"

Number assigned to a particular event (patient, visit and DCM) when the result date or time information does not match the previous date or time for the given event. Use a unique subevent number for each such assignment. Optional; default value is zero.

Example

The following example illustrates the mapping of the AEDESCTEXT control in an InForm software trial to two Oracle Clinical fields.

```
<EXTERNALMAP>

  <PATH>
    <CHAPTERREF REFNAME="PF_ALL_VISITS" />
    <PAGEREF REFNAME="AE" />
    <SECTIONREF REFNAME="AE" />
    <ITEMSETREF REFNAME="AE" />
    <ITEMREF REFNAME="AEDESC" />
    <CONTROLREF REFNAME="AEDESCTEXT" />
  </PATH>

  <ORAFIELD REFNAME="ORA1" CPEREFNAME="CPEREF1"
    DCINAME="DN1" DCMNAME="DM1" SUBSET="S1"
    QUESTIONGROUP="QG1" QUESTION="Q1"
    QUESTIONVAL="QV1" UPPERCASE="true" />
  <ORAFIELD REFNAME="ORA1" CPEREFNAME="CPEREF2"
    DCINAME="DN2" DCMNAME="DM2" SUBSET="S2"
    QUESTIONGROUP="QG2" QUESTION="Q2"
    QUESTIONVAL="QV2" UPPERCASE="false" />

</EXTERNALMAP>
```

Pageref

Purpose

The Pageref tag identifies the RefName of a *Form* (on page 122) as the location of a mapped control in the *Path* (on page 82) tag of a mapping definition. Include one Pageref tag in a RefName path defined by a *Path* (on page 82) tag.

Syntax

```
<PAGEREF
  REFNAME="name"/>
```

Attributes

REFNAME="name"

RefName of the Form in which the mapped source control occurs. Required.

Example

The following example shows the ECG form as the location where the mapped COMMONDATE control occurs.

```
<PATH>
  <CHAPTERREF REFNAME="PF_ALL_VISITS"/>
  <PAGEREF REFNAME="ECG"/>
  <SECTIONREF REFNAME="LEADECG"/>
  <ITEMSETREF REFNAME="1"/>
  <ITEMREF REFNAME="DATEASSESS"/>
  <CONTROLREF REFNAME="COMMONDATE"/>
</PATH>
```

Path

Purpose

The Path tag is a required component of a mapping definition created by the ExternalMap tag and must be the first child component tag in the ExternalMap definition. The Path tag has the following purposes:

- Used alone, with no child components, a Path tag signals the presence of mapping targets that have no corresponding control in the InForm trial. Examples of these are:
 - Columns to be created in a Customer-Defined Database that have no associated control on a CRF
 - Items common to all panels in a Clintrial protocol
- Used in conjunction with one or more child components, each of which provides the RefName of a portion of a RefName path, a Path tag specifies a control in an InForm trial. This control identifies the source data for one data mapping.

Syntax

```
<PATH>
  <CHAPTERREF attributes/>
  <PAGEREF attributes/>
  <SECTIONREF attributes/>
  <ITEMSETREF attributes/>
  <ITEMREF attributes/>
  <CONTROLREF* attributes/>
</PATH>
```

Children

Each optional child component of the Path tag specifies the RefName of one portion of the RefName path of a source control in an InForm trial:

- *Chapterref* (on page 48)—Specifies the RefName of a FormSet (Visit).
- *Pageref* (on page 81)—Specifies the RefName of a Form.
- *Sectionref* (on page 84)—Specifies the RefName of a Section.
- *Itemsetref* (on page 74)—Specifies the RefName of an ItemSet.
- *Itemref* (on page 72)—Specifies the RefName of an Item.
- *Controlref* (on page 56)—Specifies the RefName of a control. If the control is nested within another control (for example, a *SimpleControl* (on page 196) within a *PulldownControl* (on page 158)), use multiple Controlref tags to specify the path to the lowest-level control. As many as five Controlref tags are allowed.

Example

This example illustrates the use of the Path tag without child components in a mapping definition for items common to all Clintrial panels in the mapping.

```
<EXTERNALMAP>
  <PATH/>
  <CONTEXTPANEL REFNAME="MEDIKA_MAP">
    <CTITEM REFNAME="PATNUM" ITEMDATATYPE="TEXT" ISREPEAT="false"
      ISREQUIRED="true" DBFORMAT="VARCHAR2(20)" CONTEXTTYPE="1" />
    <CTITEM REFNAME="VISITID" ITEMDATATYPE="TEXT" ISREPEAT="false"
      ISREQUIRED="true" DBFORMAT="VARCHAR2(20)" CONTEXTTYPE="2" />
    <CTITEM REFNAME="FORMID" ITEMDATATYPE="TEXT" ISREPEAT="false"
      ISREQUIRED="true" DBFORMAT="VARCHAR2(20)" CONTEXTTYPE="3" />
    <CTITEM REFNAME="VISITINDEX" ITEMDATATYPE="FLOAT" ISREPEAT="true"
      ISREQUIRED="true" DBFORMAT="NUMBER(5,2)" CONTEXTTYPE="2" />
  </CONTEXTPANEL>
</EXTERNALMAP>
```

This example illustrates the use of the Path tag to define the source control for an autocode mapping.

```
<EXTERNALMAP>
  <PATH>
    <CHAPTERREF REFNAME="COMMON_CRF"/>
    <PAGEREF REFNAME="AE"/>
    <SECTIONREF REFNAME="AE"/>
    <ITEMSETREF REFNAME="AE"/>
    <ITEMREF REFNAME="AEDESC"/>
    <CONTROLREF REFNAME="AEDESCTEXT"/>
  </PATH>

  <AUTOCODE REFNAME="AEDESC1" FORMSETNAME="AEVisitRef"
    FORMNAME="AEPAGEREF" SECTIONNAME="AESectionRef"
    ITEMNAME="AEItemRef" CONTROLNAME="AEDESCCALC"/>
</EXTERNALMAP>
```

Sectionref

Purpose

The Sectionref tag enables you to:

- Include previously defined *Sections* (see "*Section*" on page 184) in the definition of a *Form* (on page 122). Include one Sectionref tag for each Section in the Form.
- Specify the RefName of a Section as the location of a mapped control in the *Path* (on page 82) tag of a mapping definition. Include one Sectionref tag in a RefName path defined by a Path tag

A Sectionref appears only as the child of a Form or Path tag in which it is included; it is not submitted as a stand-alone component.

Syntax

```
<SECTIONREF
  REFNAME="name"
  [ORDER="n"]/>
```

Attributes

REFNAME="name" RefName of the *Section* (on page 184) that the Sectionref is including in the definition of a *Form* (on page 122) or *Path* (on page 82). Required.

ORDER="n"

Sequence in which each Sectionref appears in the Form definition. Optional. If you do not specify an order, the MedML Installer utility orders the Items referred to by the Sectionrefs in the order in which you enter them.

Note: When including a Sectionref definition in a *Path* (on page 82) tag, only the REFNAME attribute is valid.

Example

This example illustrates the use of Sectionrefs in the definition of the Demographics form, which contains two *Sections* (see "*Section*" on page 184), Demographics (DEM) and Smoking History (SH). For more details about the definition of a form, see the *Form* (on page 122) topic.

```
<FORM REFNAME="DEM" TITLE="Demographics" MNEMONIC="DEM"  
FORMTYPE="CRF">  
  <SECTIONREF REFNAME="DEM"/>  
  <SECTIONREF REFNAME="SH"/>  
</FORM>
```

The following example shows the LEADECg section as the location where the mapped COMMONDATE control occurs.

```
<PATH>  
  <CHAPTERREF REFNAME="PF_ALL_VISITS"/>  
  <PAGEREF REFNAME="ECG"/>  
  <SECTIONREF REFNAME="LEADECg"/>  
  <ITEMSETREF REFNAME="1"/>  
  <ITEMREF REFNAME="DATEASSESS"/>  
  <CONTROLREF REFNAME="COMMONDATE"/>  
</PATH>
```

MedML Schema Reference

Action

Purpose

The Action tag specifies the action that results when an associated *Event* (on page 119) fires.

Syntax

```
<ACTION
  [NAME="name"]
  VALUE="Query"
  [QUERYTYPE="OPEN|CANDIDATE"]
  [LANGUAGE="name"]
  [QUERYTEXT="text"]/>
```

Attributes

NAME="name" Name used when referring to the Action in the definition of an *Event* (on page 119). The only valid name is PF_Event_Action.

VALUE="Query"

The type of action to be taken. Required. Query is the only action available.

QUERYTYPE="OPEN|CANDIDATE"

Indicates whether the query is opened as an Open Query or a Candidate Query. Optional.

LANGUAGE="name"

Language used to display the caption and control value. Optional; if you do not enter a language, English is assumed.

QUERYTEXT="text"

Text of a query created when the associated Event fires. Optional.

Example

The following example illustrates an Action that generates a query in Opened state with the text "Value Out of Range." The Action is included in the definition of an *Event* (on page 119) that also executes an *ExecutionPlan* (on page 121) called EPScript when it fires.

```
<EVENT REFNAME="Value Out of Range">
  <ACTION NAME="PF_Event_Action"
    VALUE="Query"QUERYTYPE="Open"
    QUERYTEXT="Value Out of Range"/>
  <EXECUTIONPLANREF REFNAME="EPScript"/>
</EVENT>
```

Attachruledepend

Purpose

The AttachRuleDepend tag enables you to define dependencies between a generic rule and an item, itemset or form in *AttachRuleSet* (on page 88). The attributes and child components in an AttachRuleDepend group are said to define a dependency for the context in which a rule runs.

A context can be specific to a particular FormSet, Form, Section, and ItemSet or Item (a Specific Visit context), or it can have a generic scope in either of the following ways:

- **Generic Form context**—Associates a Rule with a Form/Section/Item combination in every FormSet where the Form is defined.
- **Generic Item context**—Associates a rule with an Item in every location where the Item is defined.>

To create a generic dependency, use the special "name" values described in the definitions of the FORMSETNAME, FORMNAME, SECTIONNAME, and ITEMNAME attributes of the AttachRuleSet tag.

Note: Each dependency for a context must have the same scope as its context. For example, if a context is defined as a generic visit context, any dependencies for that context must also have generic visit attributes.

An association between two repeating forms can be a dependency of a rule context. When you define an association as a context dependency, the rule runs when a user selects or deselects the check box control that creates or dissolves an association between a specific instance of a repeating form and the other form in the association. A dependency defined for an association must be:

- Attached to a Specific Visit context
- Defined as a Trigger dependency

Syntax

```
<ATTACHRULEDEPEND
  FORMSETNAME="name"
  FORMNAME="name"
  SECTIONNAME="name"
  ITEMSETNAME="name"
  ITEMNAME="name"
  [ATTACHTYPE="APPLIED|DEPENDENCY|TRIGGER"]>
```

Attributes

FORMSETNAME="name" The RefName of the Formset for which you are specifying a dependency. To specify that the rule should be attached to the specified Form, Section, and Item in every FormSet where that combination of components occurs (Generic Form context), use "PF_ALL_VISITS" as the FORMSETNAME value. To define an association as a dependency, use the association RefName as the FORMSETNAME value. Required.

FORMNAME="name"

The Refname of the Form for which you are specifying a dependency. To specify that the rule should

be attached to the Item in every FormSet, Form, and Section in which it is defined (Generic Item context), or to define an association as a dependency, use "PF_ALL_FORMS" as the FORMNAME value. Required.

SECTIONNAME="name"

The RefName of the Section for which you are specifying a dependency. To specify that the rule should be attached to the Item in every FormSet, Form, and Section in which it is defined (Generic Item context), or to define an association as a dependency, use "PF_ALL_SECTIONS" as the SECTIONNAME value. Required.

ITEMSETNAME="name"

Name of the ItemSet containing the Item for which you are specifying a dependency. This name is specified in the RefName attribute of the ItemSet definition. Required for a dependency on an Item within an ItemSet.

ITEMNAME="name"

Name of the Item for which you are specifying a dependency. This name is specified in the RefName attribute of the Item definition. Required.

ATTACHTYPE="APPLIED | DEPENDENCY | TRIGGER"

The type of dependency you want to use. Optional; APPLIED is the default:

- **APPLIED**—Rule fires when the item or itemset is submitted after initial entry or after being changed.
- **DEPENDENCY**—Rule fires only if the item or itemset on which it is dependent contains data.
- **TRIGGER**—Rule fires even if the item or itemset on which it is dependent does not contain data. TRIGGER must be the ATTACHTYPE for an association defined as a dependency.

AttachRuleSet

Purpose

The AttachRuleSet tag associates a rule with an *Item* (on page 139) or an *ItemSet* (on page 149) on a *Form* (on page 122) or with an association between two related forms. An active Rule that is associated with an Item or ItemSet fires when the item or itemset is submitted after initial entry or after being changed. A rule attached to an association fires when a user selects or deselects the check box control that creates or dissolves an association between a specific instance of a repeating form and the other form in the association. The attributes and child components in an AttachRuleSet group are said to form the context in which a rule runs.

A context can be specific to a particular FormSet, Form, Section, and ItemSet or Item, or it can be generic in either of the following ways:

- **Generic Form context**—Associates a Rule with a Form/Section/Item combination in every FormSet where the Form is defined.
- **Generic Item context**—Associates a rule with an Item in every location where the Item is defined.

To create a generic context, use the special "name" values described in the definitions of the FORMSETNAME, FORMNAME, SECTIONNAME, and ITEMNAME attributes of the

AttachRuleSet tag.

Note: Each dependency for a context (defined with the *AttachRuleDepend* (on page 87) tag) must have the same scope as its context. For example, if a context is defined as a generic visit context, any dependencies for that context must also have generic visit attributes.

A randomization rule can have only one context.

Syntax

```
<ATTACHRULESET
  REFNAME="name"
  RULENAME="name"
  FORMSETNAME="name"
  FORMNAME="name"
  SECTIONNAME="name"
  ITEMSETNAME="name"
  ITEMNAME="name"
  [HELPTEXT="text"]
  [ATTACHTYPE="APPLIED|DEPENDENCY|TRIGGER"]
  [ACTIVE="true|false"]
  <RULEARG* attributes/>
  <ATTACHRULEDEPEND* attributes/>
  <EVENTREF attributes/>
</ATTACHRULESET>
```

Attributes

REFNAME="name" Name of the AttachRuleSet. Required. This name must be unique for each rule.

RULENAME="name"

Name of the Rule to attach to an Item. Required. This name is specified in the RefName attribute of the Rule definition.

FORMSETNAME="name"

Name of the FormSet in which the Item appears. Required. This name is specified in the RefName attribute of the FormSet definition. To specify that the rule should be attached to the specified Form, Section, and Item in every FormSet where that combination of components occurs (Generic Form context), use "PF_ALL_VISITS" as the FORMSETNAME value.

FORMNAME="name"

Name of the Form in which the Item appears. Required. This name is specified in the RefName attribute of the Form definition. To specify that the rule should be attached to the Item in every FormSet, Form, and Section in which it is defined (Generic Item context), use "PF_ALL_FORMS" as the FORMNAME value.

SECTIONNAME="name"

Name of the Section of the Form in which the Item appears. Required. This name is specified in the RefName attribute of the Section definition. To specify that the rule should be attached to the Item in every FormSet, Form, and Section in which it is defined (Generic Item context), use "PF_ALL_SECTIONS" as the SECTIONNAME value.

ITEMSETNAME="name"

Name of the ItemSet containing the Item to which the Rule is attached. This name is specified in the RefName attribute of the ItemSet definition. Required for a rule attached to an Item within an ItemSet.

ITEMNAME="name"

Name of the Item to which the Rule is attached. This name is specified in the RefName attribute of the Item definition. Required.

HELPTEXT="text"

Description of the data check performed by the rule. This description appears in the CRF help for the data item to which the rule is attached, if the CRF help includes the appropriate mapping. Optional. Helptext specified in AttachRuleSet overrides Helptext specified in the Rule.

ATTACHTYPE="APPLIED | DEPENDENCY | TRIGGER"

The type of dependency you want to use. Optional; APPLIED is the default.

- **APPLIED**—Rule fires when the item or itemset is submitted after initial entry or after being changed
- **DEPENDENCY**—Rule fires only if the item or itemset on which it is dependent contains data
- **TRIGGER**—Rule fires even if the item or itemset on which it is dependent does not contain data

ACTIVE="true | false" Indicates whether the Rule is active and will fire under the appropriate circumstances: true or false. Optional; true is the default.

Children

The definition of an AttachRuleSet can include zero or more of the following:

- **Rulearg** (on page 182) tags, each of which defines the parameters of the rule.
- **AttachRuleDepend** (on page 87) tags. Each AttachRuleDepend tag indicates the type of dependency and the item or the association the rule is dependent on.
- **Eventref** (on page 120) tags. Each Eventref tag enables you to include the definition of an **Event** (on page 119) in the definition of a **Rule** (on page 178). An Eventref specified in AttachRuleSet overrides an Eventref specified in the Rule.

Example

The following example illustrates how the definition of a Rule called "Range Checking Rule " is attached to the Weight item on the Demographics form with an AttachRuleSet definition.

```
<MEDMLDATA>

<RULE REFNAME="Range Checking Rule"
  DESCRIPTION="Range Checking Rule"
  ENABLED="true"
  SCRIPTTYPE="SERVERRULE"
  SCRIPTFILE="RangeCheck.vbs"
  HELPTEXT="Value of the item have to be in the specified range.">
  <EVENTREF REFNAME="Value Out of Range"/>
</RULE>

<ATTACHRULESET RULENAME="Range Checking Rule"
  REFNAME="attach1"
  FORMSETNAME="Visit1"
  FORMNAME="DEM"
  SECTIONNAME="DEM"
  ITEMNAME="WEIGHT"
  ATTACHTYPE="APPLIED"
  ACTIVE="true">
  <RULEARG NAME="ItemName" VALUE="0.Visit1.DEM.DEM.0.WEIGHT"
TYPE="STRING"/>
  <RULEARG NAME="Min" VALUE="90" TYPE="NUMERIC"/>
  <RULEARG NAME="Max" VALUE="275" TYPE="NUMERIC"/>
</ATTACHRULESET>

<ATTACHRULESET RULENAME="Range Checking Rule"
  REFNAME="attach2"
  FORMSETNAME="Visit1"
  FORMNAME="DEM"
  SECTIONNAME="DEM"
  ITEMNAME="HEIGHT"
  ATTACHTYPE="APPLIED"
  ACTIVE="true">
```

```

    <RULEARG NAME="ItemName" VALUE="0.Visit1.DEM.DEM.0.HEIGHT"
TYPE="STRING"/>
    <RULEARG NAME="Min" VALUE="30" TYPE="NUMERIC"/>
    <RULEARG NAME="Max" VALUE="75" TYPE="NUMERIC"/>
</ATTACHRULESET>

<RULE REFNAME="dynatest"
DESCRIPTION="Dynamic Visit Test"
ENABLED="true"
SCRIPTTYPE="SERVERCALCULATION"
SCRIPTFILE="dynaset.vbs"
HELPTEXT="Schedule Dynamic Visit"/>

<ATTACHRULESET RULENAME="dynatest"
REFNAME="attach1"
FORMSETNAME="Visit1"
FORMNAME="DEM"
SECTIONNAME="DEM"
ITEMNAME="GENDER"
ATTACHTYPE="APPLIED"
ACTIVE="true"/>

</MEDMLDATA>

```

Example 2

The following example illustrates how to pass arguments to a help text string in the associated Rule definition, defining one or more specialized help string arguments whose names begin with the string %PFHELPARG%. For example, to pass minimum and maximum range limits to a generic help text string, you could define the following arguments:

```

<RULEARG NAME="%PFHELPARG%MEASURE" VALUE="Height" TYPE="STRING"/>
<RULEARG NAME="%PFHELPARG%MIN" VALUE="48" TYPE="NUMERIC"/>
<RULEARG NAME="%PFHELPARG%MAX" VALUE="96" TYPE="NUMERIC"/>

```

These arguments could be substituted into the following HELPTEXT string in the Rule definition:

```

%PFHELPARG%MEASURE must be between %PFHELPARG%MIN and
%PFHELPARG%MAX.

```

Browser

Purpose

The Browser component specifies a supported browser type.

Note: This component is used only in the default XML files used to load the InForm database before it is delivered. The default browser component definitions should be changed only by Oracle Engineering.

Syntax

```
<BROWSER
  BROWSERNAME="name"
  BROWSERTYPE="type" />
```

Attributes

BROWSERNAME="name" Name of the browser. Required.

BROWSERTYPE="type"

Type of browser. Required. This is a calculated value defined by Oracle Development.

Example

The following example illustrates the XML code used to define the browser component for Internet Explorer version 4.01.

```
<BROWSER BROWSERNAME="IE0401" BROWSERTYPE="16897"/>
```

CalculatedControl

Purpose

The CalculatedControl tag enables you to define a control whose value is based on the value of one or more other controls.

Syntax

```
<CALCULATEDCONTROL
  REFNAME="name"
  [DESIGNNOTE="text"]
  [NAME="name"]
  [UUID="id"]
  [CAPTION="text"]
  [LANGUAGE="name"]
  [MAXLENGTH="n"]
  [CAPTIONALIGN="LEFT|RIGHT|TOP|BOTTOM"]
  [ALIGN="LEFT|CENTER|RIGHT|TOP|MIDDLE|BOTTOM"]
  [FORMAT="return_format"] /
```

```
[UNITDISPLAYTYPE="ELEMENT">  
<UNTREF attributes/>
```

Attributes

REFNAME="name" Name used when referring to the CalculatedControl in the definition of another control. Required. This name must be unique among CalculatedControls.

DESIGNNOTE="text"

Free-form text, with a maximum of 255 characters, containing any information you want to capture about the design of the component. This information is for documentation only and is not displayed. Optional.

NAME="name"

Name used when referring to the CalculatedControl in the definition of another control. Optional.

UUID="id"

Universally Unique Identifier (see "*MedML Schema*" on page 11); a string that identifies the component uniquely across all trials, trial databases, and machines. Optional.

CAPTION="text"

Text that appears with the calculated value. Optional.

LANGUAGE="name"

Language used to display the caption and control value. Optional; if you do not enter a language, English is assumed.

MAXLENGTH="n"

Maximum length allowed for the calculated value. Optional.

CAPTIONALIGN="LEFT | RIGHT | TOP | BOTTOM"

Position of the caption relative to the calculated value: Left, Right, Top, or Bottom. Optional; Left is the default.

ALIGN="LEFT | CENTER | RIGHT | TOP | MIDDLE | BOTTOM"

Alignment of the calculated value within the control: Left, Center, Right, Top, Middle, or Bottom. Optional; Left is the default.

FORMAT="return_format"

Format of the data returned by the rule that is attached to the data item defined by the CalculatedControl. Use %s to mark the location of the calculated value.

UNITDISPLAYTYPE="ELEMENT"

Type of control used to display *Units* (see "*Unit*" on page 220) included in the TextControl with a *Unitref* (on page 223) definition: Element is the only valid value.

Note: If you deactivate a calculated item from a form, you must either deactivate the calculation rule, or rename all items which are dependants for the calculation. If you remove only the calculated item itself, the rule system will fail.

Children

A CalculatedControl definition can include one *Unitref* (on page 223) definition specifying the unit to be associated with the control.

Example

The following example illustrates the definition of the control used to display the randomization string that results when a user clicks the Randomization button. The RANDOMIZATION calculated control is included in the definition of the DRUGKIT item. When the randomization rule attached to the DRUGKIT item runs, it returns the randomization string. Note that the InForm Architect application and the MedML Installer utility convert alphabetic characters in UUIDs to uppercase.

```
<CALCULATEDCONTROL REFNAME="RANDOMIZATION"
  UUID="DC2EB0BF-4F12-11d2-9319-00A0C9769A13"
  NAME="RANDOMIZATION"
  FORMAT="Drug-kit number %s has been assigned to this patient"
/>

<ITEM REFNAME="DRUGKIT"
  UUID="52AF1207-4F13-11d2-9319-00A0C9769A13"
  QUESTION="Sequence Number and Information: "
  ITEMREQUIRED="false"
  CALCULATED="true">
  <CONTROLREF REFNAME="RANDOMIZATION" />
</ITEM>
```

CheckboxControl

Purpose

The CheckBoxControl tag defines a list of check boxes from which users can select one or more options. CheckBoxControls are composed of previously defined components that you include by using *Controlrefs* (see "*Controlref*" on page 56). You can include the following types of components in a CheckBoxControl definition:

- *CalculatedControl* (on page 93)
- *CheckBoxControl* (on page 95)
- *GroupControl* (on page 134)
- *PullDownControl* (on page 158)
- *RadioControl* (on page 163)
- *SimpleControl* (on page 196)
- *TextControl* (on page 216)

Syntax

```
<CHECKBOXCONTROL
  REFNAME="name"
  [DESIGNNOTE="text"]
  [NAME="name"]
  [UUID="id"]
  [ALIGN="LEFT|CENTER|RIGHT|TOP|MIDDLE|BOTTOM"]
  [LAYOUT="VERTICAL|HORIZONTAL|NOWRAP"]
  [CAPTION="text"]
  [LANGUAGE="name"]
  [CAPTIONALIGN="LEFT|RIGHT|TOP|BOTTOM"]>
<CONTROLREF+ attributes/>

</CHECKBOXCONTROL>
```

Attributes

REFNAME="name" Name used when referring to the CheckBoxControl in the definition of another control. Required. This name must be unique among CheckBoxControls.

DESIGNNOTE="text"

Free-form text, with a maximum of 255 characters, containing any information you want to capture about the design of the component. This information is for documentation only and is not displayed. Optional.

NAME="name"

Name used when referring to the CheckBoxControl in the definition of another control. Optional.

UUID="id"

Universally Unique Identifier (see "*MedML Schema*" on page 11); a string that identifies the component uniquely across all trials, trial databases, and machines. Optional.

ALIGN="LEFT|CENTER|RIGHT|TOP|MIDDLE|BOTTOM"

Alignment of the components included with *Controlrefs* (see "*Controlref*" on page 56) within the CheckBoxControl: Left, Center, Right, Top, Middle, or Bottom. Optional; Left is the default.

LAYOUT="VERTICAL|HORIZONTAL|NOWRAP"

Orientation of the check boxes: Vertical, Horizontal, or Nowrap. Optional; Nowrap is the default. When you specify Nowrap, the radio buttons are oriented horizontally, and the buttons do not wrap to another line if a user resizes the browser window.

CAPTION="text"

Text that appears on the screen with the check box list. Optional.

LANGUAGE="name"

Language used to display the caption. Optional; English is the default.

CAPTIONALIGN="LEFT | RIGHT | TOP | BOTTOM"

Position of the caption relative to the calculated value: Left, Right, Top, or Bottom. Optional; Left is the default.

Children

A CheckBoxControl definition must include one or more *Controlref* (on page 56) definitions. Each Controlref refers to a previously defined component that identifies one item in the list of check boxes.

Note: When defining compound controls with *Controlref* (on page 56) definitions, ensure that all subordinate controls return the same data type, by defining them with the same TYPE attribute.

Additionally, when defining compound controls, note that the InForm application supports a maximum of five levels of nesting. Although five levels are supported, as a design practice, you should attempt to minimize the number of nested levels to help performance.

Example**Example 1**

This example illustrates how to create a simple CheckBoxControl by including previously defined *PFElements* (see "*PFElement*" on page 160) that have been enclosed in *SimpleControls* (see "*SimpleControl*" on page 196). The CheckBoxControl in this example is a list of patient smoking habits.

- Cigarettes
- Pipe
- Cigars
- Not Done

- 1 Define each check box component as a *PFElement* (on page 160), and enclose it in a *SimpleControl* (on page 196). Note that the "ND" component can be defined once and then reused in any control or form.
- 2

```
<PFELEMENT REFNAME="CIGARETTE" LABEL="Cigarettes" TYPE="STRING"
VALUE="CIGARETTE"/>
<PFELEMENT REFNAME="PIPE" LABEL="Pipe" TYPE="STRING"
VALUE="PIPE"/>
<PFELEMENT REFNAME="CIGAR" LABEL="Cigars" TYPE="STRING"
VALUE="CIGAR"/>
<PFELEMENT REFNAME="ND" LABEL="Not Done" TYPE="STRING"
VALUE="ND"/> <SIMPLECONTROL REFNAME="CIGARETTE"
NAME="CIGARETTE">
  <ELEMENTREF REFNAME="CIGARETTE"/>
</SIMPLECONTROL>
<SIMPLECONTROL REFNAME="PIPE" NAME="PIPE">
  <ELEMENTREF REFNAME="PIPE"/>
</SIMPLECONTROL>
<SIMPLECONTROL REFNAME="CIGAR" NAME="CIGAR">
  <ELEMENTREF REFNAME="CIGAR"/>
</SIMPLECONTROL>
```


- 3 Define CheckBoxControls for the Arms and Legs check boxes by using *Controlrefs* (see "*Controlref*" on page 56). Note that you can use the Left and Right *TextControls* (see "*TextControl*" on page 216) in each CheckBoxControl.
- 4

```
<CHECKBOXCONTROL REFNAME="ARMS" NAME="ARMS"
  LAYOUT="VERTICAL"
  CAPTION="Arms (describe): "
  CAPTIONALIGN="LEFT">
  <CONTROLREF REFNAME="LEFT" ORDER="1"/>
  <CONTROLREF REFNAME="RIGHT" ORDER="2"/>
</CHECKBOXCONTROL>
<CHECKBOXCONTROL REFNAME="LEGS" NAME="LEGS"
  LAYOUT="VERTICAL"
  CAPTION="Legs (describe): "
  CAPTIONALIGN="LEFT">
  <CONTROLREF REFNAME="LEFT" ORDER="1"/>
  <CONTROLREF REFNAME="RIGHT" ORDER="2"/>
</CHECKBOXCONTROL>
```
- 5 Define a CheckBoxControl that uses *Controlrefs* (see "*Controlref*" on page 56) to include the Arms and Legs CheckBoxControls and the Other *TextControl* (on page 216).
- 6

```
<CHECKBOXCONTROL REFNAME="MUSCLES" NAME="MUSCLES"
  LAYOUT="VERTICAL">
  <CONTROLREF REFNAME="ARMS" ORDER="1"/>
  <CONTROLREF REFNAME="LEGS" ORDER="2"/>
  <CONTROLREF REFNAME="OTHER" ORDER="3"/>
</CHECKBOXCONTROL>
```

CodeTarget (dictionary)

Purpose

The CODETARGET tag is part of the tag set that defines a coding dictionary. Each coding dictionary includes a set of code targets. A code target specifies one type of item that holds data after it has been coded.

Syntax

```
< CODETARGET
  NAME="name"/>
```

Attributes

NAME="name" Name of the code target. Required.

Example

```

<DICTIONARY TYPE="WHODD" VERSION="05Q4" CULTURE="en-US">
  <CODETARGET NAME="ATC 1.CODE"/>
  <CODETARGET NAME="ATC 1.TERM"/>
  <CODETARGET NAME="ATC 2.CODE"/>
  <CODETARGET NAME="ATC 2.TERM"/>
  <CODETARGET NAME="ATC 3.CODE"/>
  <CODETARGET NAME="ATC 3.TERM"/>
  <CODETARGET NAME="ATC 4.CODE"/>
  <CODETARGET NAME="ATC 4.TERM"/>
  <CODETARGET NAME="Preferred Name.CODE"/>
  <CODETARGET NAME="Preferred Name.TERM"/>
  <CODETARGET NAME="Ingredients.AddInfo"/>
  <CODETARGET NAME="Trade Name.CODE"/>
  <CODETARGET NAME="Trade Name.TERM"/>
  <CODETARGET NAME="Medicinal Product.CODE"/>
  <CODETARGET NAME="Medicinal Product.TERM"/>
  <CODETARGET NAME="Name Specifier.AddInfo"/>
  <CODETARGET NAME="Country of Sale.AddInfo"/>
  <CODETARGET NAME="MA Holder.AddInfo"/>
  <CODETARGET NAME="MA Holder Country.AddInfo"/>
  <CODETARGET NAME="Company.AddInfo"/>
  <CODETARGET NAME="Company Country.AddInfo"/>
  <CODETARGET NAME="ICH Med Prod ID.AddInfo"/>
  <CODETARGET NAME="Sequence Number 3.AddInfo"/>
  <CODETARGET NAME="Sequence Number 4.AddInfo"/>
  <CODETARGET NAME="MA Number.AddInfo"/>
  <CODETARGET NAME="MA Date.AddInfo"/>
  <CODETARGET NAME="MA Withdrawal Date.AddInfo"/>
  <CODETARGET NAME="Product Type.AddInfo"/>
  <CODETARGET NAME="Product Group.AddInfo"/>
  <CODETARGET NAME="Pharmaceutical Product.AddInfo"/>
  <CONTEXTITEM NAME="Route Of Administration"/>
  <CONTEXTITEM NAME="Indication"/>
  <VERBATIMTYPE NAME="MEDPROD" />
</DICTIONARY>

```

ContextItem (dictionary)

Purpose

The CONTEXTITEM tag is part of the tag set that defines a coding dictionary. A coding dictionary can include a set of context items. A context item provides additional information to enable coding of a verbatim.

Syntax

```
< CONTEXTITEM
  NAME="name"/>
```

Attributes

NAME="name" Name of the context. Required. The WHODD dictionary uses the following context item names:

- Route of Administration—The route by which the drug was administered.
- Indication—The disease or disorder for which the drug was taken.

Example

```
<DICTIONARY TYPE="WHODD" VERSION="05Q4" CULTURE="en-US">
  <CODETARGET NAME="ATC 1.CODE"/>
  <CODETARGET NAME="ATC 1.TERM"/>
  <CODETARGET NAME="ATC 2.CODE"/>
  <CODETARGET NAME="ATC 2.TERM"/>
  <CODETARGET NAME="ATC 3.CODE"/>
  <CODETARGET NAME="ATC 3.TERM"/>
  <CODETARGET NAME="ATC 4.CODE"/>
  <CODETARGET NAME="ATC 4.TERM"/>
  <CODETARGET NAME="Preferred Name.CODE"/>
  <CODETARGET NAME="Preferred Name.TERM"/>
  <CODETARGET NAME="Ingredients.AddInfo"/>
  <CODETARGET NAME="Trade Name.CODE"/>
  <CODETARGET NAME="Trade Name.TERM"/>
  <CODETARGET NAME="Medicinal Product.CODE"/>
  <CODETARGET NAME="Medicinal Product.TERM"/>
  <CODETARGET NAME="Name Specifier.AddInfo"/>
  <CODETARGET NAME="Country of Sale.AddInfo"/>
  <CODETARGET NAME="MA Holder.AddInfo"/>
  <CODETARGET NAME="MA Holder Country.AddInfo"/>
  <CODETARGET NAME="Company.AddInfo"/>
  <CODETARGET NAME="Company Country.AddInfo"/>
  <CODETARGET NAME="ICH Med Prod ID.AddInfo"/>
  <CODETARGET NAME="Sequence Number 3.AddInfo"/>
  <CODETARGET NAME="Sequence Number 4.AddInfo"/>
  <CODETARGET NAME="MA Number.AddInfo"/>
  <CODETARGET NAME="MA Date.AddInfo"/>
  <CODETARGET NAME="MA Withdrawal Date.AddInfo"/>
  <CODETARGET NAME="Product Type.AddInfo"/>
  <CODETARGET NAME="Product Group.AddInfo"/>
  <CODETARGET NAME="Pharmaceutical Product.AddInfo"/>
  <CONTEXTITEM NAME="Route Of Administration"/>
  <CONTEXTITEM NAME="Indication"/>
  <VERBATIMTYPE NAME="MEDPROD" />
</DICTIONARY>
```

Controlref

Purpose

The Controlref tag enables you to include the definition of one type of component in the definition of another component. A Controlref appears only as the child of a component in which it is included; it is not submitted as a stand-alone component.

Types of components in which you can include Controlrefs Types of components you can include by using Controlrefs

CheckBoxControl (on page 95)

GroupControl (on page 134)

Item (on page 139)

RadioControl (on page 163)

Path (on page 82)

CalculatedControl (on page 93)

CheckBoxControl (on page 95)

DateTimeControl (on page 105)

GroupControl (on page 134)

PullDownControl (on page 158)

RadioControl (on page 163)

SimpleControl (on page 196)

TextControl (on page 216)

Syntax

```
<CONTROLREF
  REFNAME="name"
  [ORDER="n"]
  [SELECTIONVALUE="text"]/>
```

Attributes

REFNAME="name" RefName of the component that the Controlref is including in the definition of a compound control. Required.

ORDER="n"

Sequence in which each Controlref appears in the compound control definition. Optional. If you do not specify an order, the MedML Installer utility orders the components referred to by the Controlrefs in the order in which you enter them.

SELECTIONVALUE="text"

Value of the compound control included by the Controlref. Optional; if you do not specify a value, when a user selects one of the controls in the compound control, the value stored in the database is a string consisting of the characters !pf! and the DBUID path of the selected control. For example, this attribute is useful for providing a value to store for a check box or radio control component labeled Other that is associated with a TextBoxControl.

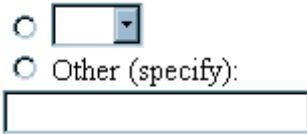
Note: When defining compound controls with Controlref or *Unitref* (on page 223) definitions, ensure that all subordinate controls return the same data type, by defining them with the same TYPE attribute.

When defining compound controls, note that the InForm application supports a maximum of five levels of nesting. Although five levels are supported, as a design practice, you should attempt to minimize the number of nested levels to help performance.

When including a Controlref definition in a *Path* (on page 82) tag, only the REFNAME attribute is valid.

Examples

This example illustrates how to create a list of radio buttons in which the first button is a pull-down list and the second is a text box. The definition of the radio button list includes the definition of each list item as a Controlref. This list will be used to capture a medication regimen:



- 1 Define the pull-down list as a set of *PFElements* (see "*PFElement*" on page 160), and include them in a *PullDownControl* (on page 158) definition by using *Elementrefs* (see "*Elementref*" on page 117).


```

2 <PFELEMENT REFNAME="QD" LABEL="QD" TYPE="STRING" VALUE="QD"/>
  <PFELEMENT REFNAME="BID" LABEL="BID" TYPE="STRING" VALUE="BID"/>
  <PFELEMENT REFNAME="TID" LABEL="TID" TYPE="STRING" VALUE="TID"/>
  <PFELEMENT REFNAME="QID" LABEL="QID" TYPE="STRING" VALUE="QID"/>
  <PULLDOWNCONTROL REFNAME="MEDREGIMEN"
    NAME="MEDREGIMEN">
    <ELEMENTREF REFNAME="QD" ORDER="1"/>
    <ELEMENTREF REFNAME="BID" ORDER="2"/>
    <ELEMENTREF REFNAME="TID" ORDER="3"/>
    <ELEMENTREF REFNAME="QID" ORDER="4"/>
  </PULLDOWNCONTROL>

```
- 3 Define the text box and its caption as a *TextControl* (on page 216).


```

4 <TEXTCONTROL REFNAME="MEDREGTEXT"
  NAME="MEDREGTEXT"
  HEIGHT="1"
  LENGTH="20"
  MAXLENGTH="20"
  DATATYPE="STRING"
  CAPTION="Other (specify): "
  CAPTIONALIGN="TOP"/>

```
- 5 Include the PullDownControl and TextControl in a *RadioControl* (on page 163) by using Controlrefs. Note the use of the SELECTIONVALUE attribute to assign the value "OTHER" to the Other (Specify): radio button.


```

6 <RADIOCONTROL REFNAME="MEDREGGROUP"
  NAME="MEDREGRADIO" LAYOUT="VERTICAL">
  <CONTROLREF REFNAME="MEDREGIMEN" ORDER="1"/>
  <CONTROLREF REFNAME="MEDREGTEXT" ORDER="2"
    SELECTIONVALUE="OTHER"/>
</RADIOCONTROL>

```

This example illustrates the use of Controlref definitions in the *Path* (on page 82) tag to identify the source control used in creating a mapping of components between the InForm database and a Customer-Defined Database.

```

<EXTERNALMAP>
  <PATH>

```



```

<CHAPTERREF REFNAME="PF_ALL_VISITS"/>
<PAGEREF REFNAME="ECG"/>
<SECTIONREF REFNAME="CHESTXRAY"/>
<ITEMSETREF REFNAME="1"/>
<ITEMREF REFNAME="INTERPRET2"/>
<CONTROLREF REFNAME="INTERPRETRADIO2"/>
<CONTROLREF REFNAME="DESCRIBETEXT"/>
</PATH>

<CDD REFNAME="CDD1" KEYTYPE="PATIENT" TARGETTABLE="t_ECG"
  TARGETKEYTYPE="PATIENTVISIT" TARGETCOLUMN="COMMONDAT1"
  TARGETCOLUMNTYPE="TEXT"/>

</EXTERNALMAP>

```

DateTimeControl

Purpose

The DateTimeControl component defines a set of pull-down lists used to select one or more of the following date or time values: Day, Month, Year, Hour, Minute, or Second.

Syntax

```

<DATETIMECONTROL
  REFNAME="name"
  [DESIGNNOTE="text"]
  [NAME="name"]
  [UUID="id"]
  [LANGUAGE="name"]
  [CAPTION="text"]
  [CAPTIONALIGN="LEFT | RIGHT | TOP | BOTTOM"]
  [ALIGN="LEFT | CENTER | RIGHT | TOP | MIDDLE | BOTTOM"]
  STARTYEAR="year"
  ENDYEAR="year"
  [DISPLAYMONTH="true | false"]
  [DISPLAYDAY="true | false"]
  [DISPLAYYEAR="true | false"]
  [DISPLAYHOUR="true | false"]
  [DISPLAYMINUTE="true | false"]
  [DISPLAYSECOND="true | false"]
  [REQUIREMONTH="true | false"]
  [REQUIREDAY="true | false"]
  [REQUIREYEAR="true | false"]
  [REQUIREHOUR="true | false"]
  [REQUIREMINUTE="true | false"]
  [REQUIRESECOND="true | false"]
  [UNKNOWNMONTH="true | false"]
  [UNKNOWNDAY="true | false"]
  [UNKNOWNYEAR="true | false"]

```

```
[UNKNOWNHOUR="true | false"]
[UNKNOWNMINUTE="true | false"]
[UNKNOWNSECOND="true | false"]
[CHECKCONSISTENT="true | false"]
[TEXTFORMAT="true | false"]/>
```

Attributes

REFNAME="name" Name used when referring to the DateTimeControl in the definition of another control. Required. This name must be unique among DateTimeControl definitions.

DESIGNNOTE="text"

Free-form text, with a maximum of 255 characters, containing any information you want to capture about the design of the component. This information is for documentation only and is not displayed. Optional.

NAME="name"

Name used when referring to the DateTimeControl in the definition of another control. Optional.

UUID="id"

Universally Unique Identifier (see "*MedML Schema*" on page 11); a string that identifies the component uniquely across all trials, trial databases, and machines. Optional.

Note: When you create a repeating *FormSet* (on page 127), most often to identify an unscheduled visit, the first form in the FormSet must contain a DateTimeControl definition that includes the following required UUID:
BD991BC0-B0A4-11D2-80E3-00A0C9AF7674
This DateTimeControl definition is illustrated in the Example section.

LANGUAGE="name"

Language used to display the question. Optional; English is the default.

CAPTION="text"

Text that appears on the screen with the date and time pull-down lists. Optional.

CAPTIONALIGN="LEFT | RIGHT | TOP | BOTTOM"

Position of the caption relative to the calculated value: Left, Right, Top, or Bottom. Optional; Left is the default.

ALIGN="LEFT | CENTER | RIGHT | TOP | MIDDLE | BOTTOM"

Alignment of the pull-down lists within the control: Left, Center, Right, Top, Middle, or Bottom. Optional; Left is the default.

STARTYEAR="year"

First year that appears in the pull-down list for the year. Required when you use the DateTimeControl to define a date.

ENDYEAR="year"

Last year that appears in the pull-down list for the year. Required when you use the DateTimeControl to define a date.

DISPLAYMONTH="true | false"

Indicates whether the control includes a pull-down list for the month. Optional; true is the default.

DISPLAYDAY="true | false"

Indicates whether the control includes a pull-down list for the day. Optional; true is the default.

DISPLAYYEAR="true | false"

Indicates whether the control includes a pull-down list for the year. Optional; true is the default.

DISPLAYHOUR="true | false"

Indicates whether the control includes a pull-down list for the hour. Optional; false is the default.

DISPLAYMINUTE="true | false"

Indicates whether the control includes a pull-down list for the minute. Optional; false is the default.

DISPLAYSECOND="true | false"

Indicates whether the control includes a pull-down list for the second. Optional; false is the default.

REQUIREMONTH="true | false"

Indicates whether the pull-down list for the month requires an entry. Optional; false is the default. This attribute is valid only if you specify true for the DISPLAYMONTH attribute.

REQUIREDAY="true | false"

Indicates whether the pull-down list for the day requires an entry. Optional; false is the default. This attribute is valid only if you specify true for the DISPLAYDAY attribute.

REQUIREYEAR="true | false"

Indicates whether the pull-down list for the year requires an entry. Optional; false is the default. This attribute is valid only if you specify true for the DISPLAYYEAR attribute.

REQUIREHOUR="true | false"

Indicates whether the pull-down list for the month requires an entry. Optional; false is the default. This attribute is valid only if you specify true for the DISPLAYHOUR attribute.

REQUIREMINUTE="true | false"

Indicates whether the pull-down list for the month requires an entry. Optional; false is the default. This attribute is valid only if you specify true for the DISPLAYMINUTE attribute.

REQUIRESECOND="true | false"

Indicates whether the pull-down list for the month requires an entry. Optional; false is the default. This attribute is valid only if you specify true for the DISPLAYSECOND attribute.

UNKNOWNMONTH="true | false"

Indicates whether the pull-down list for the month includes the selection "UNK", allowing a user to specify that the date is unknown. Optional; false is the default. This attribute is valid only if you specify true for the DISPLAYMONTH attribute.

UNKNOWNDAY="true | false"

Indicates whether the pull-down list for the day includes the selection "UNK", allowing a user to specify that the date is unknown. Optional; false is the default. This attribute is valid only if you specify true for the DISPLAYDAY attribute.

UNKNOWNYEAR="true | false"

Indicates whether the pull-down list for the year includes the selection "UNK", allowing a user to specify that the date is unknown. Optional; false is the default. This attribute is valid only if you specify true for the DISPLAYYEAR attribute.

UNKNOWNHOUR="true | false"

Indicates whether the pull-down list for the month includes the selection "UNK", allowing a user to specify that the time is unknown. Optional; false is the default. This attribute is valid only if you specify true for the DISPLAYHOUR attribute.

UNKNOWNMINUTE="true | false"

Indicates whether the pull-down list for the month includes the selection "UNK", allowing a user to specify that the time is unknown. Optional; false is the default. This attribute is valid only if you specify true for the DISPLAYMINUTE attribute.

UNKNOWNSECOND="true | false"

Indicates whether the pull-down list for the month includes the selection "UNK", allowing a user to specify that the time is unknown. Optional; false is the default. This attribute is valid only if you specify true for the DISPLAYSECOND attribute.

CHECKCONSISTENT="true | false"

Specifies whether the InForm application checks for internal consistency among the entered components of the date and time. Optional; true is the default. When the value of CHECKCONSISTENT is true:

- If any date or time component value is entered, all higher-order components displayed in the control must also have an entered, numeric value. For example, if a user enters a day value in a month/day/year date time control, the user must also enter the month and year. If the datetime control includes both date and time components and a user enters a time value, the user must also enter the date portion.
- The InForm application treats a blank control and a control with the value of UNK identically. For example, if a user enters a numeric month and enters UNK for the year, the entry generates an error.
- The consistency check applies to optional as well as required date time components; For example, if the time portion of a control is optional, the InForm application generates an error if a user enters the minutes without the hour.
- If the Year component of the datetime control is not displayed, the consistency check is disabled.

TEXTFORMAT="true | false" Specifies whether the time portion of the control is represented as a set of text boxes instead of the default set of pulldown lists. Optional; false is the default (display as pulldown lists).

Example 1

The following example illustrates a DateTimeControl definition that creates pull-down lists for the month, day, and year of a patient's birth. Month, day, and year are required. The control is included in an *Item* (on page 139) definition by using the *Controlref* (on page 56) component. Note that this example also includes a *HelpLink* (on page 137) component that specifies a link from the item to its CRF Help entry.

Date that the patient was most recently admitted to a hospital facility:

```
<DATETIMECONTROL REFNAME="dob" NAME="dob"
  UUID="40aee712-217c-11d2-a425-00a0c963e0ac"
  STARTYEAR="1932"
  ENDYEAR="1998"
  DISPLAYDAY="true" DISPLAYMONTH="true" DISPLAYYEAR="true"
  REQUIREMONTH="true" REQUIREDAY="true" REQUIREYEAR="true"/>

<ITEM REFNAME="DEMDOB" QUESTION="Date of Birth: ">
  <CONTROLREF REFNAME="dob"/>
  <HELPLINK DOCREFNAME="Study"
    BODYREFNAME="DEMHELP"
    BOOKMARK="Item2"/>
</ITEM>
```

Example 2

This example illustrates the Date of Visit *Section* (on page 184) definition that is required in the first form of every visit and in a repeating *FormSet* (on page 127). This Section definition uses a DateTimeControl with a required UUID. The control includes the month, day, year, hour, and minute of the visit and requires that at least the month and year be entered.

Date Of Visit	
1.	Date and time of visit: <input type="text"/> / <input type="text"/> / <input type="text"/> <input type="text"/> : <input type="text"/>

```
<DATETIMECONTROL REFNAME="DOV" NAME="DOV"
  UUID="BD991BC0-B0A4-11D2-80E3-00A0C9AF7674"
  STARTYEAR="1997" ENDYEAR="2004"
  DISPLAYDAY="true" DISPLAYMONTH="true"
  DISPLAYYEAR="true" DISPLAYHOUR="true"
  DISPLAYMINUTE="true"
  REQUIREMONTH="true" REQUIREYEAR="true"/>

<ITEM REFNAME="DOV" QUESTION="Date and time of visit:"
  UUID="BD991BBF-B0A4-11D2-80E3-00A0C9AF7674">
  <CONTROLREF REFNAME="DOV"/>
```

```

</ITEM>
<SECTION REFNAME="DOV" TITLE="Date Of Visit"
  UUID="BD991BBE-B0A4-11D2-80E3-00A0C9AF7674">
  <ITEMREF REFNAME="DOV" ORDER="1"/>
</SECTION>

```

Dictionary (definition)

Purpose

The **DICTIONARY** tag defines a coding dictionary

Syntax

```

<DICTIONARY
  TYPE="text"
  VERSION="text"
  CULTURE="text">
  <CODETARGET+ attributes/>
  <CONTEXTITEM* attributes/>
  <VERBATIMTYPE* attributes/>
</DICTIONARY>

```

Attributes

TYPE="text" Type of dictionary, for example, MedDRA or WHODD. Required.

VERSION="text"

Dictionary version, for example, 8.1, 05Q4. Required.

CULTURE="text"

Language and culture, for example, en-US. Required.

Note: The combination of **TYPE**, **VERSION**, and **CULTURE** values uniquely identifies a dictionary.

Children

The **DICTIONARY** tag has the following child elements:

- One or more **CODETARGET** (see "*CodeTarget (dictionary)*" on page 99) tags identifying a specific code target defined for the dictionary.
- Zero or more **CONTEXTITEM** (see "*ContextItem (dictionary)*" on page 100) tags identifying a specific context item defined for the dictionary.
- Zero or more **VERBATIMTYPE** (on page 232) tags identifying a valid verbatim type for the dictionary. The verbatim type corresponds to the Central Coding application item type.

Restrictions

The MedML Installer utility enforces the following restrictions on dictionary definition XML:

- Code target names, context item names, and verbatim type names must be unique within the dictionary
- At least one code target must be specified.

Example

```
<DICTIONARY TYPE="WHODD" VERSION="05Q4" CULTURE="en-US">
  <CODETARGET NAME="ATC 1.CODE"/>
  <CODETARGET NAME="ATC 1.TERM"/>
  <CODETARGET NAME="ATC 2.CODE"/>
  <CODETARGET NAME="ATC 2.TERM"/>
  <CODETARGET NAME="ATC 3.CODE"/>
  <CODETARGET NAME="ATC 3.TERM"/>
  <CODETARGET NAME="ATC 4.CODE"/>
  <CODETARGET NAME="ATC 4.TERM"/>
  <CODETARGET NAME="Preferred Name.CODE"/>
  <CODETARGET NAME="Preferred Name.TERM"/>
  <CODETARGET NAME="Ingredients.AddInfo"/>
  <CODETARGET NAME="Trade Name.CODE"/>
  <CODETARGET NAME="Trade Name.TERM"/>
  <CODETARGET NAME="Medicinal Product.CODE"/>
  <CODETARGET NAME="Medicinal Product.TERM"/>
  <CODETARGET NAME="Name Specifier.AddInfo"/>
  <CODETARGET NAME="Country of Sale.AddInfo"/>
  <CODETARGET NAME="MA Holder.AddInfo"/>
  <CODETARGET NAME="MA Holder Country.AddInfo"/>
  <CODETARGET NAME="Company.AddInfo"/>
  <CODETARGET NAME="Company Country.AddInfo"/>
  <CODETARGET NAME="ICH Med Prod ID.AddInfo"/>
  <CODETARGET NAME="Sequence Number 3.AddInfo"/>
  <CODETARGET NAME="Sequence Number 4.AddInfo"/>
  <CODETARGET NAME="MA Number.AddInfo"/>
  <CODETARGET NAME="MA Date.AddInfo"/>
  <CODETARGET NAME="MA Withdrawal Date.AddInfo"/>
  <CODETARGET NAME="Product Type.AddInfo"/>
  <CODETARGET NAME="Product Group.AddInfo"/>
  <CODETARGET NAME="Pharmaceutical Product.AddInfo"/>
  <CONTEXTITEM NAME="Route Of Administration"/>
  <CONTEXTITEM NAME="Indication"/>
  <VERBATIMTYPE NAME="MEDPROD" />
</DICTIONARY>
```

DocBody

Purpose

The DocBody tag defines a single .htm file that typically represents a section of a trial protocol, a set of CRF item help for a CRF or an online system help topic. DocBody tags are defined by inclusion in a *Documentation* (on page 113) definition; they do not stand alone.

Syntax

```
<DOCBODY  
  REFNAME="name"  
  FILENAME="name"  
  FILENAMEID="id"  
  [ORDER="n"]  
  [LINKS="n"]/>
```

Attributes

REFNAME="name" RefName used for referring to the DocBody file in a link specification. Required.

FILENAME="name"

Path and file name of the .htm file that the DocBody defines. Note that the path must be relative to the location from which you are running the MedML Installer utility. Either FILENAME or FILENAMEID is required. They are mutually exclusive.

FILENAMEID="id"

Either FILENAME or FILENAMEID is required. They are mutually exclusive.

ORDER="n"

Number indicating the order in which the DocBody file is displayed when a user navigates the document sequentially by clicking the paging controls in the Document or Help window. Optional; the default is the order in which the DocBody appears in the *Documentation* (on page 113) definition.

LINKS="n"

Number of links from the DocBody file to other DocBody files. This number does not include links to bookmarks within the same file. Required only if the DocBody file includes links to other DocBody files.

Example

The following example illustrates the use of DocBody tags to load the topics in one section of the InForm online Help.

```
<DOCUMENTATION REFNAME="AboutInForm"  
  DOCNAME="About the InForm application"  
  DOCTYPE="HELP"  
  HELPTEXT="Click here for an introduction to the InForm application"  
  TOC="..\XMLBase\Help\TocAboutInform.htm"
```



```

    TOCLINKS="6"
    INDEX="..\XMLBase\Help\HelpIndex.htm"
    INDEXLINKS="220">
    <DOCBODY REFNAME="WELCOME" FILENAME="..\XMLBase\Help\Welcome.htm"
LINKS="5"/>
    <DOCBODY REFNAME="FEATURES"
FILENAME="..\XMLBase\Help\InFormFeatures.htm" LINKS="4"/>
    <DOCBODY REFNAME="ROADMAP" FILENAME="..\XMLBase\Help\RoadMap.htm"
LINKS="5"/>
    <DOCBODY REFNAME="RIGHTS" FILENAME="..\XMLBase\Help\Rights.htm"
LINKS="5"/>
    <DOCBODY REFNAME="DATETIME" FILENAME="..\XMLBase\Help\DateTime.htm"
LINKS="4"/>
</DOCUMENTATION>

```

Documentation

Purpose

The Documentation tag enables you to create the definition of a document to be viewed in the Document window or the Help window. Each document represents one tab displayed in the Document or Help window. A document can contain multiple document files, each defined in a *DocBody* (on page 112) tag. In the InForm application, the following entities are represented by the Documentation tag:

- Trial protocol
- CRF help
- Sponsor documents
- Visit Calculator
- Sample Case Book
- System help

For a document to be visible in the Document or Help window, the Documentation definition must be associated with a *StudyVersion* (on page 206) in the *StudyVersionDoc* (on page 204) tag.

Syntax

```

<DOCUMENTATION
  REFNAME="name"
  DOCNAME="name"
  DOCTYPE="BOOKMARKDOC|BOOKMARKFAQDOC|VISITDOC|CRBDOC|HELP"
  [HELPTEXT="text"]
  [UUID="id"]
  [LANGUAGE="name"]
  TOC="filename"
  [TOCLINKS="n"]
  [INDEX="filename"]
  [INDEXLINKS="n"]>
  <DOCBODY+ attributes/>

```

</DOCUMENTATION>

Attributes

REFNAME="name" Name used when referring to the document in a *StudyVersionDoc* (on page 204) definition. Required.

DOCNAME="name"

Name that appears on the tag that represents the Document in the Document or Help window. Required.

DOCTYPE="BOOKMARKDOC | BOOKMARKFAQDOC | VISITDOC | CRBDOC | HELP"

Type of document; required:

- **BOOKMARKDOC**—A document that other documents can link into; used for trial protocol.
- **BOOKMARKFAQDOC**—A document that other documents can link into and that can include Frequently Asked Questions (FAQs) and help on rules; used for CRF Help.
- **VISITDOC**—The Visit Calculator screen.
- **CRBDOC**—A blank, sample set of CRF forms for the trial.
- **HELP**—A document containing system help.

HELPTEXT="text" Text that appears as floating help when a user moves the cursor over the document's tab in the Document or Help window. Optional.

UUID="id"

Universally Unique Identifier (see "*MedML Schema*" on page 11); a string that identifies the component uniquely across all trials, trial databases, and machines. Optional.

LANGUAGE="name"

Language used to display the caption. Optional; English is the default.

TOC="filename"

Path and file name of the HTM file that serves as the Table of Contents for the document. Required for **BOOKMARKDOC**, **BOOKMARKFAQDOC**, and **HELP** document types; not applicable for others. Note that the path must be relative to the location from which you are running the MedML Installer utility.

TOCLINKS="n"

Number of links from the Table of Contents file to other files in the document.

INDEX="filename"

Path and file name of the .htm file that serves as the Index for the document. Required for **HELP** document types; not applicable for others. Note that the path must be relative to the location from which you are running the MedML Installer utility.

INDEXLINKS="n"

Number of links from the Index file to other files in the document.

Children

A Document tag can include zero or more *DocBody* (on page 112) tags. Each DocBody tag defines a single .htm file that typically represents a section of a trial protocol, a set of CRF item help for a single CRF or an online system help topic. Document definitions with a VISITDOC or CRBDOC type do not include DocBody tags.

Example

The following set of Document definitions illustrates each document type. The Protocol, Trial, and Help documents include multiple .htm files, defined with *DocBody* (on page 112) tags. The Protocol and Trial documents have Tables of Contents; the Help document has both a Table of Contents and an Index.

```
<MEDMLDATA>
```

```
<DOCUMENTATION REFNAME="Protocol"
  DOCNAME="Protocol Guide"
  DOCTYPE="BOOKMARKDOC"
  HELPTTEXT="Click here to view Protocol Guide"
  TOC=".\\StudyDoc\\TOC.htm"
  TOCLINKS="25">
  <DOCBODY REFNAME= "OBJECT" FILENAME=".\\StudyDoc\\Objectives.htm"
LINKS="0"/>
  <DOCBODY REFNAME= "DESIGN" FILENAME=".\\StudyDoc\\StudyDesign.htm"
LINKS="9"/>
  <DOCBODY REFNAME= "DROPOUT" FILENAME=".\\StudyDoc\\DropoutDrug.htm"
LINKS="10"/>
  <DOCBODY REFNAME= "INCLEXCL" FILENAME=".\\StudyDoc\\InclExclCriteria.htm"
LINKS="3"/>
  <DOCBODY REFNAME= "TESCHEDULE" FILENAME=".\\StudyDoc\\TESchedule.htm"
LINKS="0"/>
  <DOCBODY REFNAME= "MEDRESTRICT" FILENAME=".\\StudyDoc\\MedRestrict.htm"
LINKS="1"/>
  <DOCBODY REFNAME= "PLACEBO" FILENAME=".\\StudyDoc\\PlaceboRunIn.htm"
LINKS="2"/>
  <DOCBODY REFNAME= "WEEK-4" FILENAME=".\\StudyDoc\\Week-4.htm"
LINKS="1"/>
  <DOCBODY REFNAME= "WEEK-2" FILENAME=".\\StudyDoc\\Week-2.htm"
LINKS="2"/>
  <DOCBODY REFNAME= "WEEK0" FILENAME=".\\StudyDoc\\Week0.htm"
LINKS="7"/>
  <DOCBODY REFNAME= "TREATMENT"
FILENAME=".\\StudyDoc\\TreatmentPhase.htm" LINKS="5"/>
  <DOCBODY REFNAME= "WEEK1" FILENAME=".\\StudyDoc\\Week1.htm"
LINKS="1"/>
  <DOCBODY REFNAME= "WEEK2" FILENAME=".\\StudyDoc\\Week2.htm"
LINKS="2"/>
  <DOCBODY REFNAME= "WEEK4" FILENAME=".\\StudyDoc\\Week4.htm"
LINKS="2"/>
  <DOCBODY REFNAME= "WEEK5" FILENAME=".\\StudyDoc\\Week5.htm"
LINKS="5"/>
```

```

    <DOCBODY REFNAME= "WEEK8" FILENAME=".\\StudyDoc\\Week8.htm"
LINKS="1"/>
</DOCUMENTATION>

<DOCUMENTATION REFNAME="Study"
  DOCNAME="CRF Help"
  DOCTYPE="BOOKMARKFAQDOC"
  HELPTEXT="Click here to view information on completing CRFs"
  TOC=".\\StudyGuide\\SG_TOC.htm"
  TOCLINKS="10">
  <DOCBODY REFNAME="SCREENHELP"
FILENAME=".\\StudyGuide\\ScreeningLog.htm"/>
  <DOCBODY REFNAME="ELIGHELP" FILENAME=".\\StudyGuide\\Eligibility.htm"/>
  <DOCBODY REFNAME="SSHHELP" FILENAME=".\\StudyGuide\\Signs_Symptoms.htm"/>
  <DOCBODY REFNAME="DEMHELP" FILENAME=".\\StudyGuide\\Demographics.htm"/>
  <DOCBODY REFNAME="VSHHELP" FILENAME=".\\StudyGuide\\VitalSigns_BP.htm"/>
  <DOCBODY REFNAME="CHESTHELP"
FILENAME=".\\StudyGuide\\ECG_Chest_XRay.htm"/>
  <DOCBODY REFNAME="PEHELP" FILENAME=".\\StudyGuide\\Physical_Exam.htm"/>
  <DOCBODY REFNAME="PEW8HELP"
FILENAME=".\\StudyGuide\\Week8_Physical_Exam.htm"/>
  <DOCBODY REFNAME="CMHELP" FILENAME=".\\StudyGuide\\Con_Meds.htm"/>
  <DOCBODY REFNAME="SCHELP"
FILENAME=".\\StudyGuide\\Study_Completion.htm"/>
</DOCUMENTATION>

<DOCUMENTATION REFNAME="Visit"
  DOCNAME="Visit Calculator"
  DOCTYPE="VISITDOC"
  HELPTEXT="Click here to view Patient Visit Calculator"
</DOCUMENTATION>

<DOCUMENTATION REFNAME="CRB"
  DOCNAME="Sample Book"
  DOCTYPE="CRBDOC"
  HELPTEXT="Click here to view Sample Case Book forms"
</DOCUMENTATION>

<DOCUMENTATION REFNAME="AboutInForm"
  DOCNAME="About InForm software"
  DOCTYPE="HELP"
  HELPTEXT="Click here for an introduction to the InForm application"
  TOC="..\XMLBase\\Help\\TocAboutInform.htm"
  TOCLINKS="6"
  INDEX="..\XMLBase\\Help\\HelpIndex.htm"
  INDEXLINKS="220">
  <DOCBODY REFNAME="WELCOME" FILENAME="..\XMLBase\\Help\\Welcome.htm"
LINKS="5"/>
  <DOCBODY REFNAME="FEATURES"
FILENAME="..\XMLBase\\Help\\InFormFeatures.htm" LINKS="4"/>
  <DOCBODY REFNAME="ROADMAP" FILENAME="..\XMLBase\\Help\\RoadMap.htm"
LINKS="5"/>

```

```
<DOCBODY REFNAME="RIGHTS" FILENAME="..\XMLBase\Help\Rights.htm"
LINKS="5"/>
<DOCBODY REFNAME="DATETIME" FILENAME="..\XMLBase\Help\DateTime.htm"
LINKS="4"/>
</DOCUMENTATION>

</MEDMLDATA>
```

Elementref

Purpose

The Elementref tag enables you to include the definition of a PFElement in the definition of the following other types of form components:

- *PullDownControl* (on page 158)
- *SimpleControl* (on page 196)

An Elementref appears only as the child of one of these components; it is not submitted as a stand-alone component.

Syntax

```
<ELEMENTREF
  REFNAME="name"
  [ORDER="n"]/>
```

Attributes

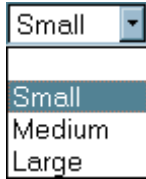
REFNAME="name" RefName of the *PFElement* (on page 160) that the Elementref is including in the *PullDownControl* (on page 158) or *SimpleControl* (on page 196). Required.

ORDER="n"

Sequence in which each PFElement appears in the PullDownControl or SimpleControl. Optional. If you do not specify an order, the MedML Installer utility orders the PFElements in the order in which you enter them.

Example

This example demonstrates how to use Elementrefs to include three previously created *PFElements* (see "*PFElement*" on page 160) that define selections describing a patient's body frame in a *PullDownControl* (on page 158) called "FRAME." In the example, the list is ordered in the same sequence in which the entries appear; if you wanted to reorder the list, you could change the Order attributes of the Elementref definitions.



- 1 Create the list items as PFElements. Note that, although the text label of each component is a string, the components are defined as integers and assigned integer values in the database.
- 2

```
<PFELEMENT REFNAME="SMALL" LABEL="Small" TYPE="INTEGER"
VALUE="1"/>
<PFELEMENT REFNAME="MEDIUM" LABEL="Medium" TYPE="INTEGER"
VALUE="2"/>
<PFELEMENT REFNAME="LARGE" LABEL="Large" TYPE="INTEGER"
VALUE="3"/>
```
- 3 By using Elementrefs, include the PFElement definitions in the FRAME PullDownControl definition.
- 4

```
<PULLDOWNCONTROL REFNAME="FRAMEPULLDOWN"
NAME="FRAME">
<ELEMENTREF REFNAME="SMALL" ORDER="1"/>
<ELEMENTREF REFNAME="MEDIUM" ORDER="2"/>
<ELEMENTREF REFNAME="LARGE" ORDER="3"/>
</PULLDOWNCONTROL>
```

Event

Purpose

The Event tag enables you to define an event that executes when its **Rule** (on page 178) fires. The definition of an Event includes the text of a query that is created when the associated Rule is not satisfied or that is closed when an answer to the query satisfies the associated Rule.

Syntax

```
<EVENT
  REFNAME="name"
  [DESIGNNOTE="text"]
  [UUID="id"]>
  <ACTION* attributes/>
  <EXECUTIONPLANREF* attributes/>
</EVENT>
```

Attributes

REFNAME="name" Name used when referring to the Event in the definition of a **Rule** (on page 178). Required. This name must be unique among Events.

DESIGNNOTE="text"

Free-form text, with a maximum of 255 characters, containing any information you want to capture about the design of the component. This information is for documentation only and is not displayed. Optional.

UUID="id"

Universally Unique Identifier (see "**MedML Schema**" on page 11); a string that identifies the component uniquely across all trials, trial databases, and machines. Required.

Children

The definition of an Event can include zero or more **Action** (on page 86) tags, each of which defines a query that is generated when the event fires. Additionally, the definition of an Event can include zero or more **ExecutionPlanref** (on page 122) tags. Each ExecutionPlanref refers to a previously defined **ExecutionPlan** (on page 121). When an Event executes, any included ExecutionPlans also execute.

Example

The following example illustrates an Event that generates a query in Opened state with the text "Value Out of Range." The Event also executes an *ExecutionPlan* (on page 121) called EPScript when it fires.

```
<EVENT REFNAME="Value Out of Range">
  <ACTION NAME="PF_Event_Action"
    VALUE="Query"
    QUERYTYPE="Open"
    QUERYTEXT="Value Out of Range"/>
  <EXECUTIONPLANREF REFNAME="EPScript"/>
</EVENT>
```

Eventref

Purpose

The Eventref tag enables you to include the definition of an *Event* (on page 119) in the definition of a *Rule* (on page 178). An Eventref appears only as the child of a Rule; it is not submitted as a stand-alone component.

Syntax

```
<EVENTREF
  REFNAME="name"/>
```

Attributes

REFNAME="name" RefName of the *Event* (on page 119) that the Eventref is including in the *Rule* (on page 178) definition. Required.

Example

In the following example, an Eventref tag includes the Value Out of Range *Event* (on page 119) definition in the definition of a *Rule* (on page 178) called Height Numeric Rule.

```
<RULE REFNAME="Height Numeric Rule"
  DESCRIPTION="Height Numeric Rule Description"
  ENABLED="true"
  SCRIPTTYPE="SERVERRULE"
  SCRIPTFILE="height.vbs"
  HELPTEXT="Height must have a numeric value between 30 AND 75.">
  <EVENTREF REFNAME="Value Out of Range"/>
</RULE>
```


ExecutionPlan

Purpose

The ExecutionPlan tag defines a script that can do the following:

- Send e-mail
- Log an entry in the NT log file

Syntax

```
<EXECUTIONPLAN
  REFNAME="name"
  [DESIGNNOTE="text"]
  SCRIPTTEXT="text"
  SCRIPTFILE="filename" />
```

Attributes

REFNAME="name"

Name used when referring to the ExecutionPlan in the definition of an *Event* (on page 119). Required. This name must be unique among ExecutionPlans.

DESIGNNOTE="text"

Free-form text, with a maximum of 255 characters, containing any information you want to capture about the design of the component. This information is for documentation only and is not displayed. Optional.

SCRIPTTEXT="text"

Text of the script. Only Visual Basic scripts are supported. Either the Scripttext or Scriptfile attribute is required.

SCRIPTFILE="filename"

Name of the file that contains the text of the script. Only Visual Basic scripts are supported. Either the Scripttext or Scriptfile attribute is required.

Example

This example illustrates an ExecutionPlan that logs a message to the NT log when the *Event* (on page 119) in which it is included fires.

```
<EXECUTIONPLAN REFNAME="EPScript"
  SCRIPTTEXT="Global.LogMessage(&quot;Query generated&quot;)" />
```

ExecutionPlanref

Purpose

The ExecutionPlanref tag enables you to include the definition of an *ExecutionPlan* (on page 121) in the definition of a *Event* (on page 119). An ExecutionPlanref appears only as the child of an Event; it is not submitted as a stand-alone component.

Syntax

```
<EXECUTIONPLANREF  
  REFNAME="name"/>
```

Attributes

REFNAME="name" RefName of the *ExecutionPlan* (on page 121) that the ExecutionPlanref is including in the *Event* (on page 119) definition. Required.

Example

In this example, the ExecutionPlanref tag includes the EPScript *ExecutionPlan* (on page 121) definition in the Value Out of Range *Event* (on page 119) definition.

```
<EXECUTIONPLAN REFNAME="EPScript"  
  SCRIPTTYPE="SERVERRULE"  
  SCRIPTTEXT="Global.LogMessage("Query generated")"/>  
  
<EVENT REFNAME="Value Out of Range">  
  <ACTION NAME="PF_Event_Action"  
    VALUE="Query" QUERYTYPE="Open"  
    QUERYTEXT="Value Out of Range"/>  
  <EXECUTIONPLANREF REFNAME="EPScript"/>  
</EVENT>
```

Form

Purpose

The Form tag enables you to create the definition of a complete form. A form consists of a group of predefined *Sections* (see "*Section*" on page 184), included by using *Sectionref* (on page 84) tags.

Syntax

```
<FORM  
  REFNAME="name"  
  [DESIGNNOTE="text"]  
  [UUID="id"]  
  TITLE="text"  
  MNEMONIC="name"  
  [NOTE="text"]  
  [LANGUAGE="name"]
```

```
[TYPE="CRF | ENROLLMENT | REGDOC | VISITREPORT | CUSTOM | CUSTOMTRIAL | CU
STOMRULES |
  CUSTOMADMIN | CUSTOMAUTH"]
[QUESTIONWIDTH="n"]
[CONTROLWIDTH="n"]
[REPEATING="true | false">
[UNIQUEKEY="true | false"]>
<SECTIONREF+ attributes/>
<HELPLINK attributes/>
<KEYITEMREF attributes/>
</FORM>
```

Attributes

REFNAME="name" Name used when referring to the Form in the definition of a *Formset* (on page 127). Required. This name must be unique among Forms.

DESIGNNOTE="text"

Free-form text, with a maximum of 255 characters, containing any information you want to capture about the design of the component. This information is for documentation only and is not displayed. Optional.

UUID="id"

Universally Unique Identifier (see "*MedML Schema*" on page 11); a string that identifies the component uniquely across all trials, trial databases, and machines. Required.

TITLE="text"

Text of the Form title, displayed at the top of the Form on the screen. Required.

MNEMONIC="name"

Abbreviation of the Form title, displayed on the tab control that a user clicks to navigate to a form. Required.

NOTE="text"

Text of a note displayed immediately below the Form title on the screen. Optional.

LANGUAGE="name"

Language used to display the Form title and note. Optional; English is the default.

TYPE="CRF | ENROLLMENT | REGDOC | VISITREPORT | CUSTOM | CUSTOMTRIAL | CUSTOMRULES | CUSTOMADMIN | CUSTOMAUTH"

Type of form. Optional; CRF is the default:

- CRF—Case Report Form. If the CRF can have multiple instances within the same visit, use the REPEATING type.
- ENROLLMENT—Screening form or Enrollment form.
- REGDOC—Form used for the Regulatory Document screen.

- VISITREPORT—Form used for the Visit Report screen.
- CUSTOM—Not used.
- CUSTOMTRIAL—Form used for screens that are integral to InForm software functionality, such as the Comment and Data Value(s) screens.
- CUSTOMRULES—Form used for rules administration screens.
- CUSTOMADMIN—Form used for administration screens such as the User, Site, and System Configuration screens.
- CUSTOMAUTH—Form used for password administration screens.

QUESTIONWIDTH="n"

Percentage of the form occupied by the question column. Optional; 50 is the default.

CONTROLWIDTH="n"

Percentage of the form occupied by the control column. Optional; 50 is the default.

REPEATING="true | false"

Indicates whether the form is repeating: true or false. Optional; false is the default. In a repeating form, multiple instances can occur within the same visit. Repeating forms can also be related in an association.

UNIQUEKEY="true | false"

Indicates whether items specified as key items for a repeating form must be unique to each instance of the form. This property applies only to repeating forms. If key items are defined as unique keys and two instances of a form are submitted in the same visit with any of the same key item values, the InForm software rejects the input of the second instance. Optional; false is the default.

Children

- Required—One or more *Sectionref* (on page 84) definitions. Each Sectionref refers to a previously defined *Section* (on page 184).
- Optional:
- *HelpLink* (on page 137)—One definition pointing to the location in a CRF Help document that is displayed in the Document window when a user clicks the underlined form name in the title bar at the top of the screen.
- *KeyItemref* (on page 152)—One or more references pointing to a *Section* (on page 184) and *Item* (on page 139) whose value is used for navigation through the instances of a repeating Form. A *KeyItemref* (on page 152) definition is only valid in the definition of a repeating Form.

Example

This example illustrates the definition of the Demographics form, which consists of two *Sections* (see "*Section*" on page 184), Demographics (DEM) and Smoking History (SH). The example assumes that individual components and controls are already defined; it shows how to define:

- *Items* (see "*Item*" on page 139) that include the components and controls
 - *Sections* (see "*Section*" on page 184) that include the Items
 - A Form that includes the Sections
- 1 Define *Items* (see "*Item*" on page 139) for the Demographics section.


```
2 <ITEM REFNAME="GENDER" QUESTION="Gender: ">
  <CONTROLREF REFNAME="GENDERRADIO"/>
</ITEM>
<ITEM REFNAME="HEIGHT" QUESTION="Height: ">
  <CONTROLREF REFNAME="HEIGHTTEXT"/>
</ITEM>
<ITEM REFNAME="WEIGHT" QUESTION="Weight: ">
  <CONTROLREF REFNAME="NUMTEXT"/>
</ITEM>
<ITEM REFNAME="FRAME" QUESTION="Frame Size: ">
  <CONTROLREF REFNAME="FRAMEPULLDOWN"/>
</ITEM>
<ITEM REFNAME="RACE" QUESTION="Race: ">
  <CONTROLREF REFNAME="RACEGROUP"/>
</ITEM>
```
 - 3 Define *Items* (see "*Item*" on page 139) for the Smoking History section.


```
4 <ITEM REFNAME="SMOKE" QUESTION="Has the patient ever smoked? ">
  <CONTROLREF REFNAME="SMOKERADIO"/>
</ITEM>
<ITEM REFNAME="EVERSMOKED" QUESTION="If the patient has ever smoked, has
the patient quit smoking? ">
  <CONTROLREF REFNAME="SMOKERADIO"/>
</ITEM>
<ITEM REFNAME="WHATSMOKED" QUESTION="If the patient currently smokes, the
patient smokes: ">
  <CONTROLREF REFNAME="SMOKECHECKBOX"/>
</ITEM>
```
 - 5 Define the Demographics and Smoking History *Sections* (see "*Section*" on page 184).


```
6 <SECTION REFNAME="DEM" TITLE="Demographics">
  <ITEMREF REFNAME="GENDER" ORDER="1"/>
  <ITEMREF REFNAME="HEIGHT" ORDER="2"/>
  <ITEMREF REFNAME="WEIGHT" ORDER="3"/>
  <ITEMREF REFNAME="FRAME" ORDER="4"/>
  <ITEMREF REFNAME="RACE" ORDER="5"/>
</SECTION>

<SECTION REFNAME="SH" TITLE="Smoking History">
  <ITEMREF REFNAME="SMOKE" ORDER="1"/>
```

```
<ITEMREF REFNAME="EVERSMOKED" ORDER="2"/>
<ITEMREF REFNAME="WHATSMOKED" ORDER="3"/>
</SECTION>
```

- 7 Define the Demographics form.
- 8

```
<FORM REFNAME="DEM" TITLE="Demographics" MNEMONIC="DEM"
TYPE="CRF">
  <SECTIONREF REFNAME="DEM"/>
  <SECTIONREF REFNAME="SH"/>
</FORM>
```

Formref

Purpose

The Formref tag enables you to include previously defined *Form* (on page 122) in the definition of a *FormSet* (on page 127). A Formref appears only as the child of a Visit in which it is included; it is not submitted as a stand-alone component.

Syntax

```
<FORMREF
  REFNAME="name"
  [DYNAMIC="true | false"]
  [ORDER="n"]
  [ALTREFNAME="name"]
  [VISITREFNAME="name"]/>
```

Attributes

REFNAME="name" RefName of the *Form* (on page 122) that the Formref is including in the definition of a *FormSet* (on page 127). Required.

DYNAMIC="true | false"

Indicates whether the form is assigned to a visit dynamically based on the value of a data item in another form. Specify true or false; false is the default.

ORDER="n"

Sequence in which each Form appears in the FormSet definition. Optional. If you do not specify an order, the MedML Installer utility orders the Forms referred to by the Formrefs in the order in which you enter them.

ALTREFNAME="name"

Identifies a new version of an old form to be presented to a patient who is participating in the trial when the *StudyVersion* (on page 206) changes. Optional. Use this attribute only in forms that have this specialized purpose. An example of the use of such a form is the collection of an additional component of data that is not time-dependent on a specific visit, such as a piece of family history.

VISITREFNAME="name"

RefName of the *FormSet* (on page 127) in which an association between *Forms* (see "*Form*" on page 122) applies. Use this attribute when the FormRef is included in a FormSet definition with a type of "Relationship".

Example

The following example shows how FormSet tags group the Demographics (DEM) and Hypertension History (HH) *Forms* (see "*Form*" on page 122) into a *FormSet* (on page 127) that make up the first visit in the trial, "Week -4." For more details, see the *FormSet* (on page 127) topic.

```
<FORMSET REFNAME="Visit1"TITLE="Week -4" TYPE="Visit"
  SCHEDULED="true"ORDER="1">
  <FORMREF REFNAME="DEM"/>
  <FORMREF REFNAME="HH"/>
  <FORMREF REFNAME="DEM"ORDER="3" ALTREFNAME="SMOKE"/>
  <FORMREF REFNAME="Preg"ORDER="4" DYNAMIC="true"/>
</FORMSET>
```

The following example illustrates how to use Formref tags to set up an association between the AE and CM forms.

```
<FORMSET REFNAME="AE_CM_RELATION"TYPE="Relationship"
  <FORMREF REFNAME="AE"VISITREFNAME="Visit2"/>
  <FORMREF REFNAME="CM"VISITREFNAME="Visit2"/>
</FORMSET>
```

FormSet**Purpose**

The FormSet tag enables you to group *Forms* (see "*Form*" on page 122) into a set of:

- CRFs that make up a visit
- Common CRF
- Screening forms
- Enrollment forms
- Monitoring forms
- Status reports
- Associated forms

Syntax

```
<FORMSET
  REFNAME="name"
  [DESIGNNOTE="text"]
  TYPE="VISIT|ENROLLMENT|SCREENING|COMMONCRF|
  MONITOR|STATUS|REGDOCS|VISITREPORTS|RELATION"
```

```

[UUID="id"]
TITLE="text"
MNEMONIC="name"
[LANGUAGE="name"]
[STARTHOUR="n"]
[ORDER="n"]
[SCHEDULED="true | false"]
[UNSCHEDULED="true | false"]
[DYNAMIC="true | false"]
[OPTIONAL="true | false"]
[REPEATING="true | false"]
[HELPTEXT="text"]>
<FORMREF+ attributes/>
</FORMSET>

```

Attributes

REFNAME="name" Name used when referring to the FormSet in the definition of a *StudyVersion* (on page 206). Required. This name must be unique among FormSets. There is a special FormSet refname, Conflict, that is required for all trials that use the InForm Unplugged application. See the example below.

DESIGNNOTE="text"

Free-form text, with a maximum of 255 characters, containing any information you want to capture about the design of the component. This information is for documentation only and is not displayed. Optional.

TYPE="VISIT | ENROLLMENT | SCREENING | COMMONCRF | MONITOR | STATUS | REGDOCS | VISITREPORTS | RELATION"

Type of FormSet. Required:

- VISIT—Set of patient visits.
- ENROLLMENT—Set of enrollment forms.
- SCREENING—Set of screening forms.
- COMMONCRF—Set of forms that occur in multiple visits and contain cumulative data; for example, a Concomitant Medication or Adverse Experience form. To define a common CRF, include the *Form* (on page 122) definition in both a COMMONCRF FormSet and in each VISIT FormSet in which the form should appear.

Warning: Once you define a common CRF by including it in a COMMONCRF FormSet, and data for any patient has been entered in it, you cannot revert the form to regular CRF status by removing it from the COMMONCRF FormSet in the *StudyVersion* (on page 206) definition. Similarly, you cannot change a CRF in a regular VISIT FormSet into a common CRF by adding it to a COMMONCRF FormSet after it contains data for any patient. If you attempt to change a StudyVersion in either of these ways, you will lose patient data.

If you need to create a regular CRF that captures the same data as an existing common CRF, create it as a separate *Form* (on page 122) definition with a different REFNAME from the common CRF, and add it to the appropriate VISIT FormSets in the StudyVersion.

- MONITOR—Set of monitoring forms: Regulatory Document checklists and Visit Reports.
- STATUS—Not currently used.
- REGDOCS—Defines the set of forms used to monitor regulatory documents.
- VISITREPORTS—Defines the set of forms used to monitor visit reports.
- RELATION—Defines a pair of associated forms. When you establish an association between forms, displaying one form results in the display of the associated form as well, with the ability to navigate from one to the other.

UUID="id"

Universally Unique Identifier (see "*MedML Schema*" on page 11); a string that identifies the component uniquely across all trials, trial databases, and machines.

Note: Some types of FormSets require specific UUIDs. When creating these types of FormSets, you must use the following UUIDs exactly as specified. Note that the InForm Architect application and the MedML Installer utility convert alphabetic characters in UUIDs to uppercase:

Enrollment	d882ce3a-0f42-11d2-a419- 00a0c963e0ac
Screening	d882ce38-0f42-11d2-a419- 00a0c963e0ac
Common	9d6bbc5d-5811-11d2-8065- 00a0c9af7674
Conflict	PF_UUID_CONFLICT_FORMSET

TITLE="text"

Title of the FormSet. Required.

MNEMONIC="name"

Abbreviation of the visit title; can be the same as the REFNAME. For FormSets that define patient visits, this abbreviation appears in the timeline at the top of the visit window. Required.

LANGUAGE="name"

Language used to display the title. Optional; English is the default.

STARTHOUR="n"

Number of hours from the start of the trial (i.e. # of hours from visit 1) used for creating the proposed time and date of this visit in the Visit Calculator. Optional.

ORDER="n"

Number that indicates the sequence in which the FormSet occurs in the visit schedule. Optional. If you do not specify a value for Order, the MedML Installer utility orders FormSets in the order in which you enter them in the *StudyVersion* (on page 206) definition.

SCHEDULED="true | false"

Indicates whether the visit is scheduled; that is, the scheduling of the date and time of the visit is specified in the trial protocol: true or false. Optional. true is the default.

UNSCHEDULED="true | false"

Indicates whether the visit is unscheduled; that is, the scheduling of the date and time of the visit is not specified in the trial protocol: true or false. Optional. false is the default.

DYNAMIC="true | false"

Indicates whether the visit is assigned dynamically to a patient based on the value of a data item in a form. Specify true or false; false is the default.

OPTIONAL="true | false"

Indicates whether a visit is optional; for internal use only. false is the default.

REPEATING="true | false"

Used with unscheduled visits to indicate whether there can be multiple unscheduled visits: true or false. Optional. false is the default.

Note: When you create a REPEATING FormSet, the following rules apply:

The first form in a repeating FormSet must not be a common CRF.

The first form in a repeating FormSet must contain a **Section** (on page 184) definition consisting of a single **Item** (on page 139) containing a **DateTimeControl** (on page 105) definition that includes the month, day, year, hour, and minute of the visit, of which at least the month and year must be required.

The Section, Item, and DateTimeControl definitions must include the following specific required **UUID** (see "**MedML Schema**" on page 11)s. Note that the InForm Architect application and the MedML Installer utility convert alphabetic characters in UUIDs to uppercase:

Section	BD991BBE-B0A4-11D2-80E3- 00A0C9AF7674
Item	BD991BBF-B0A4-11D2-80E3- 00A0C9AF7674
DateTimeControl	BD991BC0-B0A4-11D2-80E3- 00A0C9AF7674

HELPTEXT="text"

Text of the floating help associated with the FormSet. Optional.

Children

A FormSet definition must include one or more **Formref** (on page 126) definitions. Each Formref tag refers to a previously defined **Form** (on page 122).

Example

Example 1

This example illustrates a Visit FormSet for Week 2 of a trial. The FormSet is composed of three *Forms* (see "*Form*" on page 122):

- Vital Signs/Blood Pressure (VS/BP)
- Adverse Experience (AE)
- Compliance (CMP)

For the purpose of the example, the components, controls, *Items* (see "*Item*" on page 139), and *Sections* (see "*Section*" on page 184) of each form are assumed to have been previously defined.

Note: Unlike other components, FormSets are not reusable. Therefore, you define them within the definition of a StudyVersion.

- 1 Define the Vital Signs/Blood Pressure Form, which is composed of the Vital Signs (VS) and Blood Pressure (BP) Sections.
- 2

```
<FORM REFNAME="VSBP" TITLE="Vital Signs/Blood Pressure" MNEMONIC="VS/BP"
  TYPE="CRF">
  <SECTIONREF REFNAME="VS"/>
  <SECTIONREF REFNAME="BP"/>
</FORM>
```
- 3 Define the Adverse Experience Form, which has a single repeating Section called Adverse Experiences.
- 4

```
<FORM REFNAME="AE" TITLE="Adverse Experience" MNEMONIC="AE"
  TYPE="CRF">
  <SECTIONREF REFNAME="AE"/>
</FORM>
```
- 5 Define the Compliance form, which is composed of the Patient Compliance Information (PCI) and Drug Kit (DK) Sections.
- 6

```
<FORM REFNAME="CMP" TITLE="Patient Compliance" MNEMONIC="CMP"
  TYPE="CRF">
  <SECTIONREF REFNAME="PCI"/>
  <SECTIONREF REFNAME="DK"/>
</FORM>
```
- 7 Within a *StudyVersion* (on page 206) definition, define a FormSet for the Week 2 visit, including the three *Forms* (see "*Form*" on page 122) definitions by using *Formref* (on page 126) tags.
- 8

```
<STUDYVERSION
  VERSION="1"
  STUDYNAME="Hypertension Study"
  PROTOCOL="Protocol XYZZY">
  <FORMSET REFNAME="Visit1" TITLE="Week -4"          UUID="03b0d5d8- 7f2c-
11d2-a728-00a0c977c64b"
  LANGUAGE="English" TYPE="Visit"
  SCHEDULED="true" ORDER="1">
  <FORMREF REFNAME="DEM"/>
```

```

    <FORMREF REFNAME="FH"/>
    <FORMREF REFNAME="DOC"/>
  </FORMSET>
  <FORMSET REFNAME="Visit2" TITLE="Week -2"
    TYPE="Visit" SCHEDULED="true">
    <FORMREF REFNAME="VS"/>
  </FORMSET>
  <FORMSET REFNAME="Visit3" TITLE="Week 0"
    TYPE="Visit" SCHEDULED="true">
    <FORMREF REFNAME="VS"/>
  </FORMSET>
  <FORMSET REFNAME="WK2" TITLE="Week 2" UUID="F4699051-69E2-11d2-8FB5-
00A0C977C66A">
    TYPE="VISIT"
    SCHEDULED="true">
    <FORMREF REFNAME="VSBP"/>
    <FORMREF REFNAME="AE"/>
    <FORMREF REFNAME="CMP"/>
  </FORMSET>
  <FORMSET REFNAME="screen" TITLE="Screening Log"
    UUID="d882ce38-0f42-11d2-a419- 00a0c963e0ac"
    TYPE="SCREENING" SCHEDULED="false">
    <FORMREF REFNAME="screen"/>
  </FORMSET>
  <FORMSET REFNAME="enroll" TITLE="Enrollment"
    UUID="d882ce3a-0f42-11d2-a419- 00a0c963e0ac"
    TYPE="ENROLLMENT" SCHEDULED="false">
    <FORMREF REFNAME="enroll"/>
  </FORMSET>
</STUDYVERSION>

```

Example 2

This example illustrates how to include a common CRF in a StudyVersion. In this example, the AE form is a common form and is included in a CommonCRF FormSet and in the Visit FormSets for Week -2, Week 0, and the Unscheduled visit. In the CommonCRF FormSet, the UUID specified in the example and the CommonCRF TYPE attribute are required.

Additionally, this example includes an association called "CMtoAE" linking the AE and CM forms in the CommonCRF FormSet.

```

<STUDYVERSION
  VERSION="2"
  STUDYNAME="Hypertension Study"
  PROTOCOL="Protocol XYZZY">

  <FORMSET REFNAME="CommonCRF" TITLE="Common CRF"
    UUID="9d6bbc5d-5811-11d2-8065-00a0c9af7674" TYPE="COMMONCRF"
    SCHEDULED="false">
    <FORMREF REFNAME="AE" />
  </FORMSET>

  <FORMSET REFNAME="Visit1" TITLE="Week -4" LANGUAGE="English"
    UUID="03b0d5d8-7f2c-11d2-a728- 00a0c977c64b"

```

```

TYPE="Visit"
SCHEDULED="true" ORDER="1">
<FORMREF REFNAME="DEM" ORDER="3"/>
<FORMREF REFNAME="inclusion" ORDER="1"/>
<FORMREF REFNAME="exclusion" ORDER="2"/>
</FORMSET>

<FORMSET REFNAME="Visit2" TITLE="Week -2" TYPE="Visit"
SCHEDULED="true">
<FORMREF REFNAME="VS" />
<FORMREF REFNAME="AE" />
</FORMSET>

<FORMSET REFNAME="Visit3" TITLE="Week 0" TYPE="Visit"
SCHEDULED="true">
<FORMREF REFNAME="VS" />
<FORMREF REFNAME="AE" />
</FORMSET>

<FORMSET REFNAME="Visit4" TITLE="Week 2" TYPE="Visit"
SCHEDULED="true" UUID="F4699051-69E2-11d2- 8FB5-00A0C977C66A">
<FORMREF REFNAME="VS1" ORDER="1"/>
<FORMREF REFNAME="VS2" ORDER="2"/>
<FORMREF REFNAME="VS3" ORDER="3"/>
<FORMREF REFNAME="VS4" ORDER="4"/>
<FORMREF REFNAME="VS5" ORDER="5"/>
<FORMREF REFNAME="VS6" ORDER="6"/>
<FORMREF REFNAME="VS7" ORDER="7"/>
<FORMREF REFNAME="VS8" ORDER="8"/>
<FORMREF REFNAME="VS9" ORDER="9"/>
<FORMREF REFNAME="VS10" ORDER="10"/>
<FORMREF REFNAME="VS11" ORDER="11"/>
<FORMREF REFNAME="VS12" ORDER="12"/>
</FORMSET>

<FORMSET REFNAME="Visit5" TITLE="Unsched" TYPE="Visit"
SCHEDULED="false" REPEATING="true">
<FORMREF REFNAME="DEM" ORDER="1"/>
<FORMREF REFNAME="VS" ORDER="2"/>
<FORMREF REFNAME="AE" />
</FORMSET>

<FORMSET REFNAME="conflict" TITLE="Conflict" MNEMONIC="Conflict"
UUID="PF_UUID_CONFLICT_FORMSET"
TYPE="Visit" UNSCHEDULED="false" REPEATING="true"
DYNAMIC="true">
<FORMREF REFNAME="SYS_CONFLICT"/>
</FORMSET>

<FORMSET REFNAME="CMtoAE" TYPE="RELATION"
TITLE="ConMed to AE relationship" MNEMONIC="AECMRelation"
STARTHOUR="0" ORDER="17">
<FORMREF REFNAME="AE" ORDER="1" VISITREFNAME="CommonCRF"/>
<FORMREF REFNAME="CM" ORDER="2" VISITREFNAME="CommonCRF"/>

```

```
</FORMSET>
```

```
</STUDYVERSION>
```

GroupControl

Purpose

The GroupControl enables you to treat a set of components as a single component for the purpose of including it in a nested control or item definition. You can include the following types of components in a GroupControl definition:

- *CalculatedControl* (on page 93)
- *CheckBoxControl* (on page 95)
- *GroupControl* (on page 134)
- *PullDownControl* (on page 158)
- *RadioControl* (on page 163)
- *SimpleControl* (on page 196)
- *TextControl* (on page 216)

Syntax

```
<GROUPCONTROL  
  REFNAME="name"  
  [NAME="name"]  
  [UUID="id"]  
  [DESIGNNOTE="text"]  
  [ALIGN="LEFT|CENTER|RIGHT|TOP|MIDDLE|BOTTOM"]  
  [LAYOUT="VERTICAL|HORIZONTAL|NOWRAP"]  
  [CAPTION="text"]  
  [LANGUAGE="name"]  
  [CAPTIONALIGN="LEFT|RIGHT|TOP|BOTTOM"]>  
  <CONTROLREF+ attributes/>  
</GROUPCONTROL>
```

Attributes

REFNAME="name" Name used when referring to the GroupControl in the definition of another control. Required. This name must be unique among GroupControls.

NAME="name"

Name used when referring to the GroupControl in the definition of another control. Optional.

UUID="id"

Universally Unique Identifier (see "*MedML Schema*" on page 11); a string that identifies the component uniquely across all trials, trial databases, and machines. Optional.

DESIGNNOTE="text"

Free-form text, with a maximum of 255 characters, containing any information you want to capture about the design of the component. This information is for documentation only and is not displayed. Optional.

ALIGN="LEFT | CENTER | RIGHT | TOP | MIDDLE | BOTTOM"

Alignment of the components included with *Controlrefs* (see "*Controlref*" on page 56) within the GroupControl: Left, Center, Right, Top, Middle, or Bottom. Optional; Left is the default.

LAYOUT="VERTICAL | HORIZONTAL | NOWRAP"

Orientation of the check boxes: Vertical, Horizontal, or Nowrap. Optional; Nowrap is the default. When you specify Nowrap, the radio buttons are oriented horizontally, and the buttons do not wrap to another line if a user resizes the browser window.

CAPTION="text"

Text that appears on the screen with the group. Optional.

LANGUAGE="name"

Language used to display the caption. Optional; English is the default.

CAPTIONALIGN="LEFT | RIGHT | TOP | BOTTOM"

Position of the caption relative to the calculated value: Left, Right, Top, or Bottom. Optional; Left is the default.

Children

A GroupControl definition must include one or more *Controlref* (on page 56) definitions. Each Controlref refers to a previously defined component that identifies one item in the group.

Note: When defining complex controls with *Controlref* (on page 56) or *Unitref* (on page 223) definitions, ensure that all subordinate controls return the same data type, by defining them with the same TYPE attribute.

Additionally, when defining compound controls, note that the InForm application supports a maximum of five levels of nesting. Although five levels are supported, as a design practice, you should attempt to minimize the number of nested levels to help performance.

Example

The GroupControl in this example combines two *TextControls* (see "*TextControl*" on page 216) to capture the two components of a blood pressure measurement.

/ mm Hg

- 1 Define one *TextControl* (on page 216) for the first text box and the slash separating the text boxes, and define a second TextControl for the second text box and the units.

- 2

```
<TEXTCONTROL REFNAME="BP1" NAME="BP1"
  CAPTION="/" CAPTIONALIGN="RIGHT"
  HEIGHT="1"
  LENGTH="5" MAXLENGTH="5"
  DATATYPE="INTEGER"/>
<TEXTCONTROL REFNAME="BP2" NAME="BP2"
  CAPTION="mm Hg" CAPTIONALIGN="RIGHT"
  HEIGHT="1"
  LENGTH="5" MAXLENGTH="5"
  DATATYPE="INTEGER"/>
```
- 3 Define a GroupControl that uses *Controlrefs* (see "*Controlref*" on page 56) to include the two *TextControls* (see "*TextControl*" on page 216).
- 4

```
<GROUPCONTROL REFNAME="BPGRP" NAME="BPGRP"
  LAYOUT="HORIZONTAL">
  <CONTROLREF REFNAME="BP1" ORDER="1"/>
  <CONTROLREF REFNAME="BP2" ORDER="2"/>
</GROUPCONTROL>
```

GroupType

Purpose

The GroupType tag loads the group types supported in the InForm application: ItemGroup, ManagerUser, Query, Reporting, and Signature. These GroupTypes are preloaded with your InForm installation using XML that is internal to Oracle. Do not make any changes to this XML.

- 4 `<HELPLINK DOCREFNAME="Study"
BODYREFNAME="DEMHELP"
BOOKMARK="Item2"/>`
- 5 In the .xml file that defines the CRF form, include the HelpLink as a child of the appropriate *Item* (on page 139):
- 6 `<ITEM REFNAME="DEMDOB" QUESTION="Date of Birth: ">
 <CONTROLREF REFNAME="dob"/>
 <HELPLINK DOCREFNAME="Study"
 BODYREFNAME="DEMHELP"
 BOOKMARK="Item2"/>
</ITEM>`

HTMLTemplate

Purpose

The HTMLTemplate component defines a set of HTML code used internally by the InForm application when it renders forms.

Note: This component is used only in the default .xml files used to load the InForm database before it is delivered. The default HTML template definitions should be changed only by Oracle Development.

Syntax

```
<HTMLTEMPLATE
  TEMPLATENAME="name"
  BROWSERNAME | BROWSER="name"
  RESOURCEUUID="id"
  [DESCRIPTION="text"]
  [UUID="id"/>
```

Attributes

TEMPLATENAME="name" Name of the HTML template. Required.

BROWSERNAME | BROWSER="name"

Name of the browser with which to use the template; use either BROWSERNAME or BROWSER attribute. This name matches the BROWSERNAME attribute of the Browser component that defines the browser with which the template is used. Required.

RESOURCEUUID="id"

Universally Unique Identifier for the template; a string that identifies the component uniquely across all trials, trial databases, and machines. Required.

DESCRIPTION="text"

Description of the HTML template. Optional.

UUID="id"

Universally Unique Identifier (see "*MedML Schema*" on page 11); a string that identifies the component uniquely across all trials, trial databases, and machines. Optional.

Examples

The following example shows the generated XML code used to load a piece of HTML code used in rendering enrollment forms:

```
<HTMLTEMPLATE
  TEMPLATENAME="PF_HTML_ENROLLOVERFRAMESET"
  BROWSERNAME="NS0000"
  RESOURCEUUID="PF_RESOURCE_NS0000_ENROLLOVERFRAMESET"/>
```

Item

Purpose

The Item tag enables you to combine previously defined controls into a data point on a form. An item consists of a question or prompt that requests data from a user and one or more controls that provide the means to enter the data.

Syntax

```
<ITEM
  REFNAME="name"
  [DESIGNNOTE="text"]
  QUESTION="text"
  [LABEL="text"]
  [UUID="id"]
  [LANGUAGE="name"]
  [CALCULATED="true | false"]
  [ITEMREQUIRED="true | false"]
  [SDVREQUIRED="true | false"]
  [DISPLAYOVERRIDE="READONLY | EDITABLE | HIDDEN"]>
  <CONTROLREF attributes/>
  <HELPLINK attributes/>
</ITEM>
```

Attributes

REFNAME="name" Name used when referring to the Item in the definition of another control. Required. This name must be unique among Items.

DESIGNNOTE="text"

Free-form text, with a maximum of 255 characters, containing any information you want to capture about the design of the component. This information is for documentation only and is not displayed. Optional.

QUESTION="text"

Text of the question or prompt that instructs the user to enter a specific data item. Required.

Note: You can use HTML tags to embed formatting instructions such as bold, italic, or color specifications in the text of questions.

LABEL="text"

Short form of the question, for use as column headings with Items that appear in *ItemSets* (see "*ItemSet*" on page 149). Optional; the text specified in the QUESTION attribute is the default. Use of this attribute is strongly recommended in ItemSets.

UUID="id"

Universally Unique Identifier (see "*MedML Schema*" on page 11); a string that identifies the component uniquely across all trials, trial databases, and machines. Optional.

Note: When you create a repeating *FormSet* (on page 127), most often to identify an unscheduled visit, the first form in the FormSet must contain an Item with a *DateTimeControl* (on page 105) definition. The Item definition must include the following required UUID:

BD991BBF-B0A4-11D2-80E3-00A0C9AF7674

This Item definition is illustrated in the Example section.

LANGUAGE="name"

Language used to display the question. Optional; English is the default.

CALCULATED="true | false"

Indicates whether the item contains a CalculatedControl: true or false. Optional; false is the default. If you specify that the Calculated attribute is true, the item is unnumbered, to set it off from items that require user entry.

ITEMREQUIRED="true | false"

Indicates whether the item is required for data entry on the form to be complete: true or false. Optional; true is the default. If you specify false, the item shows up gray on the CRF after submission to indicate that entry is not required. Not supported in ItemSets.

SDVREQUIRED="true | false"

Indicates whether the item requires source verification: true or false. Optional; true is the default. If you specify false, the item shows up gray on the SV view of the CRF to indicate that source verification is not required.

DISPLAYOVERRIDE="READONLY | EDITABLE | HIDDEN"

Overrides the default item display mode, which is determined by the rights in a user's rights group. Values are:

- **READONLY**—Users can view but not enter data in the item, regardless of their rights group.
- **EDITABLE**—Users can enter or edit data in the item, regardless of their rights group.
- **HIDDEN**—Users cannot see the item, regardless of their rights group. On the CRF, item rows are renumbered and the item does not appear to be on the form.
- **Note:** If the item is in an *Item Group* (on page 144), the display setting for that *Item Group* (on page 144) takes precedence over the DISPLAYOVERRIDE setting.

Children

An Item definition must include one *Controlref* (on page 56) definitions. Each Controlref refers to a previously defined component that identifies a data entry option. Optionally, an Item definition can include one *HelpLink* (on page 137) definition pointing to the location in a CRF Help document that is displayed in the Document window when a user clicks the underlined item description.

Examples**Example 1**

This example illustrates the sequence for creating an item that requests information about a patient's smoking habits.

If the patient currently smokes, the patient smokes:

<input type="checkbox"/>	Cigarettes
<input type="checkbox"/>	Pipe
<input type="checkbox"/>	Cigars
<input type="checkbox"/>	Not Done

- 1 Create *PFElement* (on page 160) definitions for each check box. Note that commonly used components such as the Not Done component can be defined elsewhere and used by reference.
- 2

```
<PFELEMENT REFNAME="CIGARETTE" LABEL="Cigarettes" TYPE="STRING" VALUE="cigarette"/>
<PFELEMENT REFNAME="PIPE" LABEL="Pipe" TYPE="STRING" VALUE="pipe"/>
<PFELEMENT REFNAME="CIGAR" LABEL="Cigars" TYPE="STRING" VALUE="cigar"/>
<PFELEMENT REFNAME="ND" LABEL="Not Done" TYPE="STRING" Value="nd"/>
```
- 3 Enclose each *PFElement* (on page 160) definition in a *SimpleControl* (on page 196) by using an *Elementref* (on page 117) tag.

- 4 <SIMPLECONTROL REFNAME="CIGARETTE" NAME="WHATSMOKED">
 <ELEMENTREF REFNAME="CIGARETTE"/>
 </SIMPLECONTROL>
 <SIMPLECONTROL REFNAME="PIPE" NAME="WHATSMOKED">
 <ELEMENTREF REFNAME="PIPE"/>
 </SIMPLECONTROL>
 <SIMPLECONTROL REFNAME="CIGAR" NAME="WHATSMOKED">
 <ELEMENTREF REFNAME="CIGAR"/>
 </SIMPLECONTROL><SIMPLECONTROL REFNAME="ND"
 NAME="WHATSMOKED">
 <ELEMENTREF REFNAME="ND"/>
 </SIMPLECONTROL>
- 5 Use the *SimpleControl* (on page 196) definitions in a *CheckBoxControl* (on page 95) by using *Controlref* (on page 56) tags.
- 6 <CHECKBOXCONTROL REFNAME="SMOKECHECKBOX"
 NAME="SMOKECHECKBOX" LAYOUT="VERTICAL">
 <CONTROLREF REFNAME="CIGARETTE" ORDER="1"/>
 <CONTROLREF REFNAME="PIPE" ORDER="2"/>
 <CONTROLREF REFNAME="CIGAR" ORDER="3"/>
 <CONTROLREF REFNAME="ND" ORDER="4"/>
 </CHECKBOXCONTROL>
- 7 Use the *CheckBoxControl* (on page 95) definition in the definition of the Item by using a *Controlref* (on page 56) tag.
- 8 <ITEM REFNAME="WHATSMOKED" QUESTION="If the patient currently smokes, the patient smokes: ">
 <CONTROLREF REFNAME="SMOKECHECKBOX"/>
 </ITEM>
- 9 Add a *HelpLink* (on page 137) definition to point to the CRF Help for the item.
- <ITEM REFNAME="WHATSMOKED" QUESTION="If the patient currently smokes, the patient smokes: ">
 <CONTROLREF REFNAME="SMOKECHECKBOX"/>
 <HELPLINK DOCREFNAME="Study" BODYREFNAME="DEMHELP"
 BOOKMARK="Item9"/>
 </ITEM>

Example 2

This example illustrates the Date of Visit *Section* (on page 184) definition that is required in the first form of a repeating *FormSet* (on page 127). This Section definition uses an Item definition with a *DateTimeControl* (on page 105). The Section, Item, and DateTimeControl definitions all include specific required UUIDs.

Date Of Visit		
1.	Date and time of visit:	<input type="text"/> / <input type="text"/> / <input type="text"/> <input type="text"/> : <input type="text"/>

```
<DATETIMECONTROL REFNAME="DOV" NAME="DOV"
  UUID="BD991BC0-B0A4-11D2-80E3-00A0C9AF7674"
  STARTYEAR="1997" ENDYEAR="2004"
  DISPLAYDAY="true" DISPLAYMONTH="true"
  DISPLAYYEAR="true" DISPLAYHOUR="true"
  DISPLAYMINUTE="true"
  REQUIREMONTH="true" REQUIREYEAR="true"/>
```

```
<ITEM REFNAME="DOV" QUESTION="Date and time of visit:"
  UUID="BD991BBF-B0A4-11D2-80E3-00A0C9AF7674">
  <CONTROLREF REFNAME="DOV"/>
</ITEM>
```

```
<SECTION REFNAME="DOV" TITLE="Date Of Visit"
  UUID="BD991BBE-B0A4-11D2-80E3-00A0C9AF7674">
  <ITEMREF REFNAME="DOV" ORDER="1"/>
</SECTION>
```

ItemGroup

Purpose

The ItemGroup tag allows you to combine previously defined Items into a group for the purpose of setting their display option within a *RightsGroup* (on page 176).

Syntax

```
<ITEMGROUP  
  GROUPNAME="name"  
  [GROUPDESCRIPTION="name"]  
  <ITEMREF attributes/>  
</ITEMGROUP>
```

Attributes

GROUPNAME="name" Name used when referring to the ItemGroup in the definition of an *ItemGroupRef* (on page 145). Required. This name must be unique among ItemGroups.

GROUPDESCRIPTION="name"

Description of the ItemGroup; optional.

Children

An ItemGroup description can include any number of *Itemref* (on page 72) descriptions.

Each *Itemref* (on page 72) tag identifies one *Item* (on page 139) to be included in the ItemGroup.

Example

The following example illustrates the inclusion of the RACE and GENDER items in the CRC_Hidden ItemGroup.

```
<ITEMGROUP  
  GROUPNAME="CRC_Hidden"  
  GROUPDESCRIPTION="Items hidden from CRC">  
  <ITEMREF REFNAME="GENDER"/>  
  <ITEMREF REFNAME="RACE"/>  
</ITEMGROUP>
```


ItemGroupref

Purpose

The ItemGroupref tag enables you to include a previously defined *Item Group* (on page 144) within the definition of a *RightsGroup* (on page 176). An ItemGroupref appears only as the child of a *RightsGroup* (on page 176) in which it is included; it is not submitted as a stand-alone component.

Syntax

```
<ITEMGROUPEF  
  REFNAME="name"  
  DISPLAYOVERRIDE="READONLY | EDITABLE | HIDDEN"/>
```

Attributes

REFNAME="name" Name used when referring to the ItemGroup in the definition of a *RightsGroup* (on page 176). Required. This name must be unique among ItemGroups.

DISPLAYOVERRIDE="READONLY | EDITABLE | HIDDEN"

Overrides the item-level display mode of the items in the *Item Group* (on page 144). Required. The item-level display mode is set with the DISPLAYOVERRIDE attribute of the *Item* (on page 139) definition. Values are:

- **READONLY**—Users can view but not enter data in the item, regardless of their rights group and the item-level display mode.
- **EDITABLE**—Users can enter or edit data in the item, regardless of their rights group and the item-level display mode.
- **HIDDEN**—Users cannot see the item, regardless of their rights group and the item-level display mode. On the CRF, item rows are renumbered and the item does not appear to be on the form.

Example

The following example illustrates the inclusion of the CRC_Hidden *Item Group* (on page 144) in the CRC RG *Rights Group* (on page 176).

```
<ITEMGROUP
  GROUPNAME="CRC_Hidden"
  GROUPEDESCRIPTION="Items hidden from CRC">
  <ITEMREF REFNAME="GENDER"/>
  <ITEMREF REFNAME="RACE"/>
</ITEMGROUP>

<RIGHTSGROUP GROUPNAME="CRC RG">
  <RIGHTREF RIGHT="Print" />
  <RIGHTREF RIGHT="Custom Reports" />
  <RIGHTREF RIGHT="Canned Reports" />
  <RIGHTREF RIGHT="Enroll Patients" />
  <RIGHTREF RIGHT="View CRF" />
  <RIGHTREF RIGHT="Enter Data into a CRF" />
  <RIGHTREF RIGHT="Edit Data on a CRF" />
  <RIGHTREF RIGHT="Enter Comments into a CRF" />
  <RIGHTREF RIGHT="Mark a CRF as Ready for SV" />
  <RIGHTREF RIGHT="Mark and unmark a CRB as Ready for SV" />
  <RIGHTREF RIGHT="Answer Query" />
  <RIGHTREF RIGHT="View Connections" />
  <RIGHTREF RIGHT="View Default Connection" />
  <RIGHTREF RIGHT="Synchronize New Data" />
  <USERREF USERNAME="crc" />
  <ITEMGROUPREF REFNAME="CRC_Hidden" DISPLAYOVERRIDE="HIDDEN"/>
</RIGHTSGROUP>
```

Itemref

Purpose

The Itemref tag enables you to include previously defined Items in the definition of a section of a form.

The Itemref tag enables you to:

- Include previously defined *Items* (see "*Item*" on page 139) in the definition of a *Section* (on page 184), *ItemSet* (on page 149), or *ItemGroup* (on page 144). Include one Itemref tag for each Item in the Section, ItemSet, or ItemGroup definition.
- Specify the RefName of an Item as the location of a mapped control in the *Path* (on page 82) tag of a mapping definition. Include one Itemref tag in a RefName path defined by a Path tag.

An Itemref appears only as the child of a *Section* (on page 184) or *Path* (on page 82) in which it is included; it is not submitted as a stand-alone component.

Syntax

```
<ITEMREF
  REFNAME="name"
  [KEYITEM="true | false"]
  [UNIQUEKEY="true | false"]
  [ORDER="n"]/>
```

Attributes

REFNAME="name"

RefName of the *Item* (on page 139) that the Itemref is including in the definition of a *Section* (on page 184) or *Path* (on page 82). Required.

KEYITEM="true | false"

Indicates whether the item that the Itemref is including is a Key Item in an *ItemSet* (on page 149) definition: true or false. Optional; false is the default. The values of Key Items are displayed in a pull-down list in the InForm application to enable users to navigate to a specific instance of the itemset. Only items with text, numeric, or date/time values can be key items. Items with compound controls, including *CheckBoxControls* (see "*CheckboxControl*" on page 95), *GroupControls* (see "*GroupControl*" on page 134), *RadioControls* (see "*RadioControl*" on page 163), or *PulldownControls* (see "*PulldownControl*" on page 158), cannot be key items.

Note: You can specify Key Items for repeating forms as well as for itemsets. To specify repeating form Key Items, use the *KeyItemref* (on page 152) tag. Items that are defined as Key Items in an itemset cannot also be Key Items in a repeating form.

UNIQUEKEY="true | false"

Indicates whether the Key Item must be unique within the *FormSet* (on page 127), *Form* (on page 122), and *ItemSet* (on page 149): true or false. Optional; false is the default. If a Key Item is defined as unique and two rows of an itemset are submitted in the same visit and form with the same Key Item value, the InForm application rejects the input of the second instance.

ORDER="n"

Sequence in which each Itemref appears in the *Section* (on page 184) definition. Optional. If you do not specify an order, the MedML Installer utility orders the Items referred to by the Itemrefs in the order in which you enter them.

Note: When including an Itemref definition in a *Path* (on page 82) tag, only the REFNAME attribute is valid.

Examples

This example illustrates the use of Itemrefs in the definition of a *Section* (on page 184) called Duration of Hypertension, which contains two *Items* (see "*Item*" on page 139). For a complete illustration of the steps needed to create a Section definition, see the *Section* (on page 184) topic.

```
<SECTION REFNAME="HYPERTENSION"
  TITLE="Duration of Hypertension">
  <ITEMREF REFNAME="HTYESNO" ORDER="1"/>
  <ITEMREF REFNAME="HTDURATION" ORDER="2"/>
</SECTION>
```

The following example shows the use of an Itemref definition in a *Path* (on page 82) tag. The CLIN Item is the location where the mapped TESTTEXT control occurs.

```
<PATH>
  <CHAPTERREF REFNAME="PF_ALL_VISITS"/>
  <PAGEREF REFNAME="LAB"/>
  <SECTIONREF REFNAME="LAB"/>
  <ITEMSETREF REFNAME="LAB"/>
  <ITEMREF REFNAME="CLIN"/>
  <CONTROLREF REFNAME="CCGROUP"/>
  <CONTROLREF REFNAME="TESTTEXT"/>
</PATH>
```

The following example shows how three key items designated as a unique key combination are included in an ItemSet definition with Itemref tags.

```
<ITEMSET REFNAME="itsDOSE" ITEMREQUIRED="true"
  SDVREQUIRED="true" UNIQUEKEY="true">
  <ITEMREF REFNAME="itmDOSEFromDate" ORDER="1"
    UNIQUEKEY="true" KEYITEM="true"/>
  <ITEMREF REFNAME="itmDOSEToDate" ORDER="2"
    UNIQUEKEY="true" KEYITEM="true"/>
  <ITEMREF REFNAME="itmDOSEBlisterpack" ORDER="3"
    UNIQUEKEY="true" KEYITEM="true"/>
  <ITEMREF REFNAME="itmDOSETotalCaps" ORDER="4"/>
  <ITEMREF REFNAME="itmDOSENum" ORDER="5"/>
  <ITEMREF REFNAME="itmDOSEMissed" ORDER="6"/>
  <ITEMREF REFNAME="itmDOSEReasChange" ORDER="7"/>
  <ITEMREF REFNAME="Item_InclComments" ORDER="8"/>
</ITEMSET>
```

ItemSet

Purpose

The ItemSet component enables you to define a set of items to be used in a form with repeating, additive information; for example, an Adverse Event or Concomitant Medication form. Each set of items appears in a single row of the form.

The following rules apply to creating ItemSets:

- You can add an ItemSet component only to a **Section** (on page 184) definition with the REPEATING attribute set to true.
- Only one ItemSet definition can appear in a REPEATING Section definition.
- Regular, non-repeating **Item** (on page 139) definitions cannot appear in a REPEATING Section definition.

Syntax

```
<ITEMSET
  REFNAME="name"
  [DESIGNNOTE="text"]
  [UUID="id"]
  [ITEMREQUIRED="true | false"]
  [SDVREQUIRED="true | false"]
  [DISPLAYOVERRIDE="READONLY | EDITABLE | HIDDEN"]
  [UNIQUEKEY="true | false"]>
  <ITEMREF+ attributes/>
</ITEMSET>
```

Attributes

REFNAME="name" Name used when referring to the ItemSet in the definition of another control. Required. This name must be unique among ItemSet definitions.

DESIGNNOTE="text"

Free-form text, with a maximum of 255 characters, containing any information you want to capture about the design of the component. This information is for documentation only and is not displayed. Optional.

UUID="id"

Universally Unique Identifier (see "*MedML Schema*" on page 11); a string that identifies the component uniquely across all trials, trial databases, and machines. Optional.

ITEMREQUIRED="true | false"

Indicates whether the item is required for data entry on the form to be complete: true or false. Optional; true is the default. If you specify false, the item shows up gray on the CRF after submission to indicate that entry is not required.

SDVREQUIRED="true | false"

Indicates whether the item requires source verification: true or false. Optional; true is the default. If you specify false, the item shows up gray on the SV view of the CRF to indicate that source verification is not required.

DISPLAYOVERRIDE="READONLY | EDITABLE | HIDDEN"

Overrides the default display itemset mode, which is determined by the rights in a user's rights group. Values are:

- **READONLY**—Users can view but not enter data in the itemset, regardless of their rights group.
- **EDITABLE**— Users can edit data in the itemset, regardless of their rights group.
- **HIDDEN**—Users cannot see the itemset, regardless of their rights group. On the CRF, itemset rows are renumbered and the itemset does not appear to be on the form.

UNIQUEKEY="true | false" Indicates whether the key formed by all included Items that are defined as Key Items is a unique key: true or false. Optional; false is the default. If you specify true, and two rows of an itemset are submitted in the same visit and form with the any Key Items having the same value, the InForm application rejects the input of the second instance.

Children

An ItemSet definition must include one or more *Itemref* (on page 72) definitions. Each Itemref refers to a previously defined *Item* (on page 139).

Example

This example illustrates how to add a set of repeating items that describe an adverse event:

- 1 Create definitions of controls for each *Item* (on page 139) in the ItemSet. Use the QUESTION attribute of the Item definition to specify the text of the question as it appears on the form where users enter data; use the LABEL attribute to specify the text of the column heading in the tabular form where data are displayed.
- 2


```
<TEXTCONTROL REFNAME="SYMPTOMS"
  NAME="SYMPTOMS"
  HEIGHT="4"
  LENGTH="40"
  MAXLENGTH="40"
  DATATYPE="STRING"/> <DATETIMECONTROL REFNAME="AESTARTDATE"
NAME="AE_STARTDATE"
  UUID="AE-STARTDATE-UUID"
  STARTYEAR="1998"
  ENDYEAR="2000"
  DISPLAYMONTH="true"
  DISPLAYDAY="true"
  DISPLAYYEAR="true"
  REQUIREMONTH="true"
  REQUIREDAY="true"
  REQUIREYEAR="true"/> <DATETIMECONTROL REFNAME="AEENDDATE"
NAME="AE_ENDDATE"
  UUID="AE-ENDDATE-UUID"
  STARTYEAR="1998"
  ENDYEAR="2000"
  DISPLAYMONTH="true"
  DISPLAYDAY="true"
  DISPLAYYEAR="true"
  REQUIREMONTH="true"
  REQUIREDAY="true"
  REQUIREYEAR="true"/> <ITEM REFNAME="AE_SYMPTOM"
  QUESTION="Describe the patient's symptoms: "
  LABEL="Symptoms">
  <CONTROLREF REFNAME="SYMPTOMS"/>
</ITEM> <ITEM REFNAME="AE_START"
  QUESTION="Date that symptoms first appeared: "
  LABEL="Start Date">
  <CONTROLREF REFNAME="SYMPTOMS"/>
</ITEM> <ITEM REFNAME="AE_END"
  QUESTION="Date that symptoms stopped: "
  LABEL="End Date">
  <CONTROLREF REFNAME="SYMPTOMS"/>
</ITEM>
```
- 3 Group the *Item* (on page 139) definitions in an ItemSet definition by using *Itemref*(on page 72) definitions.

- 4

```
<ITEMSET REFNAME="AE">
  <ITEMREF REFNAME="AE_SYMPTOM"/>
  <ITEMREF REFNAME="AE_START"/>
  <ITEMREF REFNAME="AE_END"/>
</ITEMSET>
```
- 5 Create a **Section** (on page 184) definition that includes the ItemSet by using an **Itemref** (on page 72) definition. The REPEATING attribute of the Section definition must have a value of true.
- 6

```
<SECTION REFNAME="AESECTION"
  TITLE="Adverse Events"
  REPEATING="true">
  <ITEMREF REFNAME="AE"/>
</SECTION>
```

KeyItemref

Purpose

The KeyItemref tag enables you to identify an **Item** (on page 139) in a repeating **Form** (on page 122) as a Key Item for the purpose of navigating through instances of the **Form** (on page 122). The values of Key Items are displayed in a pull-down list in the InForm application to enable users to navigate to a specific instance of the form. A KeyItemref appears only as the child of a **Form** (on page 122) in which it is included; it is not submitted as a stand-alone component.

Only items with the following types of controls can be key items and are included in the selection list: date time controls, pulldown lists, radio groups, and text boxes. Items defined as radio groups cannot be key items if they contain any type of nested compound controls: check box controls, group controls, or other radio groups.

Note: You can specify Key Items for itemsets as well as for repeating forms. To specify itemset Key Items, use the KEYITEM attribute of an **Itemref** (on page 72) tag that includes an **Item** (on page 139) tag in an **ItemSet** (on page 149) definition. Items that are defined as Key Items in an itemset cannot also be Key Items in a repeating form.

Syntax

```
<KEYITEMREF
  SECTIONREFNAME="name"
  ITEMREFNAME="name"
  [ORDER="n"]
  [UNIQUEKEY="true|false"]/>
```

Attributes

SECTIONREFNAME="name" RefName of the **Section** (on page 184) where the **Item** (on page 139) that the KeyItemref is including occurs. Required.

ITEMREFNAME="name"

RefName of the **Item** (on page 139) that the KeyItemref is including in the definition of repeating **Form** (on page 122). Required.

ORDER="n"

Sequence in which each Key Item appears in the navigational pull-down list on the repeating *Form* (on page 122). Optional. If you do not specify an order, the MedML Installer utility orders the Key Items in the order in which they appear in the *Form* (on page 122) definition.

UNIQUEKEY="true | false"

Indicates whether the Key Item must be unique within the FormSet and Form: true or false. Optional; false is the default. If a Key Item is defined as unique and two instances of a form are submitted in the same visit with the same Key Item value, the InForm application rejects the input of the second instance.

Example

The following example of a portion of a Form definition illustrates the inclusion of the DATEASSESS item as a Key Item in the VS section of the VS form.

```
<SECTION REFNAME="VS" TITLE="Vital Signs">
  <ITEMREF REFNAME="DATEASSESS" ORDER="1"/>
  <ITEMREF REFNAME="WEIGHT" ORDER="2"/>
  <ITEMREF REFNAME="TEMPTEXT" ORDER="3"/>
  <ITEMREF REFNAME="BPREADING" ORDER="4"/>
  <ITEMREF REFNAME="PULSERATE" ORDER="5"/>
  <ITEMREF REFNAME="PULSERHYTHM" ORDER="6"/>
  <ITEMREF REFNAME="RESPRATE" ORDER="7"/>
</SECTION>

<FORM REFNAME="VS" TITLE="Vital Signs" MNEMONIC="VS"
  QUESTIONWIDTH="25">
  <SECTIONREF REFNAME="VS"/>
  <KEYITEMREF SECTIONREFNAME="VS" ITEMREFNAME="DATEASSESS"/>
</FORM>
```

Language**Purpose**

For future development of localization capability.

Note: Only English is supported in InForm application. For trials that run with the InForm application, Oracle recommends leaving the Language attribute out of definitions and letting the MedML Installer utility default to English.

Syntax

```
<LANGUAGE  
  LANGUAGENAME="name"  
  LCID="id"/>
```

Attributes

LANGUAGENAME="name" Not currently used.

LCID="id"

Not currently used.

ManagerRef

Purpose

The Managerref tag specifies the name of an InForm application user who is the manager of a group of other users in a group defined by the *ManagerUserGroup* (on page 155) tag. The Managerref tag is used specifically to designate a CRA manager for the purpose of generating CRA views of reports.

Syntax

```
<MANAGERREF  
  USERNAME="name"/>
```

Attributes

USERNAME="name" Login name of the InForm application user who serves in a managing capacity in the *ManagerUserGroup* (on page 155) in which the Managerref definition is included.

Example

The following example illustrates the inclusion of the Managerref tag in the definition of the ReportList *ManagerUserGroup* (on page 155).

```
<MANAGERUSERGROUP GROUPNAME="ReportList"  
  GROUPEDESCRIPTION="CRA List for Report Viewing"  
  UUID="PF_CRA_REPORT_GROUP">  
  <MANAGERREF USERNAME="sm"/>  
  <USERREF USERNAME="cra"/>  
  <USERREF USERNAME="Apu"/>  
</MANAGERUSERGROUP>
```

ManagerUserGroup

Purpose

The ManagerUserGroup tag enables you to create a group of InForm application users that serves a specific purpose. For the InForm application, the ManagerUserGroup tag is used specifically to define a group of CRAs with a manager for the purpose of generating CRA views of reports.

Note that the only way to remove a user from a group is through the Admin function of the InForm application. You cannot remove a user by running the MedML Installer utility.

Syntax

```
<MANAGERUSERGROUP
  GROUPNAME="name"
  [GROUPDESCRIPTION="text"]
  [UUID="id"]>
  <MANAGERREF* attributes/>
  <USERREF* attributes/>
</MANAGERUSERGROUP>
```

Attributes

GROUPNAME="name" Name of the ManagerUserGroup. To specify the CRA report group, use the name ReportList. Required.

GROUPDESCRIPTION="text"

Description of the ManagerUserGroup. Optional.

UUID="id"

Universally Unique Identifier (see "*MedML Schema*" on page 11); a string that identifies the component uniquely across all trials, trial databases, and machines. Optional.

Note: The definition of the ReportList ManagerUserGroup requires the following specific UUID: PF_CRA_REPORT_GROUP. Note that the InForm Architect application and MedML Installer utility convert alphabetic characters in UUIDs to uppercase.

Children

A ManagerUserGroup definition can include zero or more references to the InForm application users in *Managerref* (on page 154) or *Userref* (on page 232) tags. To include users in a ManagerUserGroup definition, use the Userref tag; to designate a user as the manager of the group, use the Managerref tag. In the ManagerUserGroup used for generating CRA report views, users included with the Userref tag appear in the list of CRAs for whom CRA report views can be generated, and users included with the Managerref tag can generate those views.

Important: If there are any sites for whom no users identified with *Userrefs* (see "*Userref*" on page 232) are CRAs, data from these sites does not show up in the reports that list by CRA. Furthermore, any reports that can be filtered by CRA (generally the monthly reports) do not display data for sites that have no assigned CRA.

If two CRAs in the ManagerUserGroup are associated with a single site, then the data from that site shows up as the responsibility of both of these CRAs; thus, the totals may appear different from those that appear when a user generates the site version of the report. For this reason, it is a good idea to list only the more senior CRA in the ManagerUserGroup definition.

Example

The following example defines the ManagerUserGroup used for CRA report viewing.

```
<MANAGERUSERGROUP GROUPNAME="ReportList"
  GROUPDESCRIPTION="CRA List for Report Viewing"
  UUID="PF_CRA_REPORT_GROUP">
  <MANAGERREF USERNAME="sm"/>
  <USERREF USERNAME="cra"/>
  <USERREF USERNAME="Apu"/>
</MANAGERUSERGROUP>
```

MedMLData

Purpose

The MedMLData tag is the first and last required tag in an .xml file that defines trial definition components. It wraps all other elements, signaling to the MedML Installer utility that the enclosed tags consist of trial definition components.

Syntax

```
<MEDMLDATA xmlns="version_information">
  [xmlns="version_information"]
  <child_components+ attributes/>
</MEDMLDATA>
```

Version of the XML contained in the file; required for XML intended to be processed by the version 4.0 MedML Installer utility. The required version text for InForm version 4.0 XML is "PhaseForward-MedML- Inform4".

Children

The child components of the MedMLData tag define components of a trial. They include components for defining visits, forms, rules and events, resources, and trial administration data. For information about the MedML tags for defining these components, see the topics in the MedML Schema Reference section of this help.

Example

The following example illustrates the use of the MedMLData tag as a wrapper for the MedML tags that define the DOV form.

```
<MEDMLDATA xmlns="PhaseForward-MedML-Inform4">

  <DATETIMECONTROL REFNAME="DOV" CAPTIONALIGN="LEFT" ALIGN="LEFT"
    STARTYEAR="2002" ENDEYEAR="2008" LANGUAGE="English"
    REQUIREMINUTE="FALSE" REQUIRESECOND="FALSE" REQUIREDAY="TRUE"
    UNKNOWNDAY="FALSE" DISPLAYYEAR="TRUE" REQUIREMONTH="TRUE"
    UNKNOWNYEAR="FALSE" UNKNOWNSECOND="FALSE" DISPLAYDAY="TRUE"
    REQUIREYEAR="TRUE" UNKNOWNMINUTE="FALSE" DISPLAYMINUTE="TRUE"
    DISPLAYSECOND="FALSE" REQUIREHOUR="FALSE"
    UUID="BD991BC0-B0A4-11D2-80E3-00A0C9AF7674"
    DISPLAYHOUR="TRUE" UNKNOWNMONTH="FALSE" UNKNOWNHOUR="FALSE"
    CHECKCONSISTENT="TRUE" DISPLAYMONTH="TRUE"/>

  <ITEM REFNAME="DOV" LANGUAGE="English"
    SDVREQUIRED="TRUE"
    QUESTION="Date and Time of Visit"
    UUID="BD991BBF-B0A4-11D2-80E3-00A0C9AF7674"
    ITEMREQUIRED="TRUE"
    CALCULATED="FALSE"
    LABEL="Date and Time of Visit">
    <CONTROLREF REFNAME="DOV" />
  </ITEM>

  <SECTION REFNAME="DOV" REPEATING="FALSE"
    LANGUAGE="English"
    TITLE="Visit Date and Time"
    UUID="BD991BBE-B0A4-11D2-80E3-00A0C9AF7674">
    <ITEMREF REFNAME="DOV" />
  </SECTION>

  <FORM REFNAME="frmDOV" LANGUAGE="English"
    TITLE="DATE OF VISIT"
    CONTROLWIDTH="65"
    TYPE="CRF"
    MNEMONIC="DOV"
    QUESTIONWIDTH="35">
    <SECTIONREF REFNAME="DOV" ORDER="1"/>
  </FORM>

</MEDMLDATA>
```

PullDownControl

Purpose

The PullDownControl tag defines a drop-down list box in which a user can select a single item. PullDownControls are composed of previously defined *PFElements* (see "*PFElement*" on page 160) that you include by using *Elementrefs* (see "*Elementref*" on page 117). PullDownControls can also contain previously defined *Units* (see "*Unit*" on page 220) that you include by using *Unitrefs* (see "*Unitref*" on page 223).

Syntax

```
<PULLDOWNCONTROL
  REFNAME="name"
  [DESIGNNOTE="text"]
  [NAME="name"]
  [UUID="id"]
  [ALIGN="LEFT|CENTER|RIGHT|TOP|MIDDLE|BOTTOM"]
  [CAPTION="text"]
  [LANGUAGE="name"]
  [CAPTIONALIGN="LEFT|RIGHT|TOP|BOTTOM"]
  [UNITDISPLAYTYPE="ELEMENT">
  <ELEMENTREF+ attributes/>
  <UNTREF attributes/>
</PULLDOWNCONTROL>
```

Attributes

REFNAME="name" Name used when referring to the PullDownControl in the definition of another control. Required. This name must be unique among PullDownControls.

DESIGNNOTE="text"

Free-form text, with a maximum of 255 characters, containing any information you want to capture about the design of the component. This information is for documentation only and is not displayed. Optional.

NAME="name"

Name used when referring to the PullDownControl in the definition of another control. Optional.

UUID="id"

Universally Unique Identifier (see "*MedML Schema*" on page 11); a string that identifies the component uniquely across all trials, trial databases, and machines. Optional.

ALIGN="LEFT|CENTER|RIGHT|TOP|MIDDLE|BOTTOM"

Alignment of the PFElements within the control: Left, Center, Right, Top, Middle, or Bottom. Optional; Left is the default.

CAPTION="text"

Text that appears on the screen with the drop-down list. Optional.

LANGUAGE="name"

Language used to display the caption. Optional; English is the default.

CAPTIONALIGN="LEFT | RIGHT | TOP | BOTTOM"

Position of the caption relative to the calculated value: Left, Right, Top, or Bottom. Optional; Left is the default.

UNITDISPLAYTYPE="ELEMENT"

Type of control used to display *Units* (see "*Unit*" on page 220) included in the PullDownControl with a *Unitref* (on page 223) definition: Element is the only valid value.

Children

A PullDownControl definition must include one or more *Elementref* (on page 117) definitions. Each *Elementref* refers to a previously defined *PFElement* (on page 160) that identifies one item in the drop-down list. Additionally, a PullDownControl definition can include one *Unitref* (on page 223) definition, which refers to a previously defined *Unit* (on page 220).

Note: When defining compound controls with *Elementref* (on page 117) or *Unitref* (on page 223) definitions, ensure that all subordinate controls return the same data type, by defining them with the same **TYPE** attribute.

Additionally, when defining compound controls, note that the InForm application supports a maximum of five levels of nesting. Although five levels are supported, as a design practice, you should attempt to minimize the number of nested levels to help performance.

Example

The PullDownControl in this example consists of a list of reasons for prescribing medication.



To create the list:

- 1 Define a *PFElement* (on page 160) for each medication reason.
- 2

```
<PFELEMENT REFNAME="ADVERSE" LABEL="Adverse Experience"
TYPE="INTEGER" VALUE="1"/>
<PFELEMENT REFNAME="CHRONIC" LABEL="Chronic Illness" TYPE="INTEGER"
VALUE="2"/>
<PFELEMENT REFNAME="HORMONE" LABEL="Hormone Replacement"
TYPE="INTEGER" VALUE="3"/>
<PFELEMENT REFNAME="PROPHYL" LABEL="Prophylaxis" TYPE="INTEGER"
VALUE="4"/>
```
- 3 Define a PullDownControl that includes each *PFElement* definition by using an *Elementref* (on page 117).

```
4 <PULLDOWNCONTROL REFNAME="MEDREASON"
  NAME="MEDREASON">
  <ELEMENTREF REFNAME="ADVERSE" ORDER="1"/>
  <ELEMENTREF REFNAME="CHRONIC" ORDER="2"/>
  <ELEMENTREF REFNAME="HORMONE" ORDER="3"/>
  <ELEMENTREF REFNAME="PROPHYL" ORDER="4"/>
</PULLDOWNCONTROL>
```

PFElement

Purpose

The PFElement tag defines an component such as an item in a pull-down list or an option in a list of radio buttons. PFElement is the lowest-level form component you can define. To use a PFElement, you must enclose it in the definition of another control such as a *PullDownControl* (on page 158) or a *SimpleControl* (on page 196).

Note: Unlike other form components, PFElements definitions cannot have multiple versions in the database. If you attempt to submit a revised version of an component with the same REFNAME as an existing PFElement, the MedML Installer utility issues a warning and does not update the component definition.

Syntax

```
<PFELEMENT
  REFNAME="name"
  [DESIGNNOTE="text"]
  LABEL="text"
  [LANGUAGE="name"]
  TYPE="STRING|INTEGER|FLOAT"
  VALUE="n"
  [UUID="id"]/>
```

Attributes

REFNAME="name" Name used when referring to the PFElement in the definition of another control. Required. This name must be unique among PFElements. A RefName can have a maximum of 63 characters.

DESIGNNOTE="text"

Free-form text, with a maximum of 255 characters, containing any information you want to capture about the design of the component. This information is for documentation only and is not displayed. Optional.

LABEL="text"

Text with which the PFElement is displayed; that is, the way it appears in a list. A Label can have a maximum of 255 characters. Required.

LANGUAGE="name"

Language used to display the label. Optional; English is the default.

TYPE="STRING|INTEGER|FLOAT"

String, Integer, or Float. Required; String is the default.

Note: All of the PFElements used in a single multiple selection control (*CheckBoxControl* (on page 95)) must have the same TYPE attribute.

VALUE="n"

Value returned when the component is selected in the database. This can be different from the RefName and Label.

UUID="id"

Universally Unique Identifier (see "*MedML Schema*" on page 11); a string that identifies the component uniquely across all trials, trial databases, and machines. Optional.

Example

The following set of tags defines two PFElements that can later be used as components in a list.

```
<PFELEMENT REFNAME="Y" LABEL="Yes"TYPE="STRING" VALUE="Y"/>
<PFELEMENT REFNAME="N" LABEL="No" TYPE="STRING"VALUE="N"/>
```

QueryGroup

Purpose

A QueryGroup specifies a set of users who can close queries initiated by other members of the same group.

Note that the only way to remove a user from a group is through the Admin function of the InForm application. You cannot remove a user by running the MedML Installer utility.

Syntax

```
<QUERYGROUP
  GROUPNAME="name"
  [GROUPDESCRIPTION="text"]
  [UUID="id"]>
  <USERREF* attributes/>
</QUERYGROUP>
```

Attributes

GROUPNAME="name"

Name of the QueryGroup. Required.

GROUPDESCRIPTION="text"

Text describing the QueryGroup. Optional.

UUID="id"

Universally Unique Identifier (see "*MedML Schema*" on page 11); a string that identifies the component uniquely across all trials, trial databases, and machines. Optional.

Children

A QueryGroup definition can include zero or more *Userref* (on page 232) definitions. Each Userref refers to a previously created *User* (on page 227) definition that identifies one InForm user.

Example

In the following QueryGroup, users named Krusty and George can each close queries initiated by the other user.

```
<QUERYGROUP GROUPNAME="CRA Query">
  <USERREF USERNAME="Krusty"/>
  <USERREF USERNAME="George"/>
</QUERYGROUP>
```

RadioControl

Purpose

The RadioControl tag defines a list of radio buttons from which users must select one option. RadioControls are composed of previously defined components that you include by using *Controlrefs* (see "*Controlref*" on page 56). You can include the following types of components in a RadioControl definition:

- *CalculatedControl* (on page 93)
- *CheckBoxControl* (on page 95)
- *GroupControl* (on page 134)
- *PullDownControl* (on page 158)
- *RadioControl* (on page 163)
- *SimpleControl* (on page 196)
- *TextControl* (on page 216)

Syntax

```
<RADIOCONTROL
  REFNAME="name"
  [DESIGNNOTE="text"]
  [NAME="name"]
  [UUID="id"]
  [ALIGN="LEFT|CENTER|RIGHT|TOP|MIDDLE|BOTTOM"]
  [LAYOUT="VERTICAL|HORIZONTAL|NOWRAP"]
  [CAPTION="text"]
  [LANGUAGE="name"]
  [CAPTIONALIGN="LEFT|RIGHT|TOP|BOTTOM"]>
  <CONTROLREF+ attributes/>
</RADIOCONTROL>
```

Attributes

REFNAME="name" Name used when referring to the RadioControl in the definition of another control. Required. This name must be unique among RadioControls.

DESIGNNOTE="text"

Free-form text, with a maximum of 255 characters, containing any information you want to capture about the design of the component. This information is for documentation only and is not displayed. Optional.

NAME="name"

Name used when referring to the RadioControl in the definition of another control. Optional.

UUID="id"

Universally Unique Identifier (see "*MedML Schema*" on page 11); a string that identifies the component uniquely across all trials, trial databases, and machines. Optional.

ALIGN="LEFT | CENTER | RIGHT | TOP | MIDDLE | BOTTOM"

Alignment of the components included with *Controlrefs* (see "*Controlref*" on page 56) within the RadioControl: Left, Center, Right, Top, Middle, or Bottom. Optional; Left is the default.

LAYOUT="VERTICAL | HORIZONTAL | NOWRAP"

Orientation of the radio buttons: Vertical, Horizontal, or Nowrap. Optional; Nowrap is the default. When you specify Nowrap, the radio buttons are oriented horizontally, and the buttons do not wrap to another line if a user resizes the browser window.

CAPTION="text"

Text that appears on the screen with the radio button list. Optional.

LANGUAGE="name"

Language used to display the caption. Optional; English is the default.

CAPTIONALIGN="LEFT | RIGHT | TOP | BOTTOM"

Position of the caption relative to the RadioControl: Left, Right, Top, or Bottom. Optional; Left is the default.

Children

A RadioControl definition must include one or more *Controlref* (on page 56) definitions. Each Controlref refers to a previously defined component that identifies one item in the list of radio buttons.

Note: When defining complex controls with *Controlref* (on page 56) definitions, ensure that all subordinate controls return the same data type, by defining them with the same TYPE attribute.

Additionally, when defining compound controls, note that the InForm application supports a maximum of five levels of nesting. Although five levels are supported, as a design practice, you should attempt to minimize the number of nested levels to help performance.

Examples

Example 1

This example continues the *SimpleControl* (on page 196) example, which demonstrates how to define a set of *PFElements* (see "*PFElement*" on page 160) for specifying gender and wrap them in SimpleControls. The current example shows how to use *Controlrefs* (see "*Controlref*" on page 56) to include the previously defined SimpleControls in a RadioControl:

Male Female

- 1 Create PFElements that define selections for "Male" and "Female."
- 2

```
<PFELEMENT REFNAME="MALE" LABEL="Male" TYPE="STRING"
VALUE="MElement"/>
<PFELEMENT REFNAME="FEMALE" LABEL="Female" TYPE="STRING"
VALUE="FElement"/>
```
- 3 Create SimpleControls that use an *Elementref* (on page 117) to include the definition of each PFElement.
- 4

```
<SIMPLECONTROL REFNAME="MALE" NAME="MALE">
  <ELEMENTREF REFNAME="MALE"/>
</SIMPLECONTROL>
<SIMPLECONTROL REFNAME="FEMALE" NAME="FEMALE">
  <ELEMENTREF REFNAME="FEMALE"/>
</SIMPLECONTROL>
```
- 5 Using Controlrefs to include each SimpleControl, create a RadioControl.
- 6

```
<RADIOCONTROL REFNAME="GENDER"
NAME="GENDERRADIO"
LAYOUT="HORIZONTAL">
  <CONTROLREF REFNAME="MALE" ORDER="1"/>
  <CONTROLREF REFNAME="FEMALE" ORDER="2"/>
</RADIOCONTROL>
```

Example 2

This example illustrates the process of creating a list of radio buttons that include several different types of controls, some simple and some nested. The example defines a radio button list of reasons why a screened trial patient did not complete the trial.

Did not meet criteria for randomization
 Adverse experience (other than adverse laboratory experience)
 Adverse laboratory experience
 Patient decision
 Reason:
 Physician Decision
 Reason:
 Sponsor request
 Reason:
 Death (Sponsor must be notified and Serious Adverse Experience Report completed)
 Date of Death:
 Lost to follow-up
 Other (describe):

Components of the list:

- This list includes several items that you can simply define as *PFElements* (see "*PFElement*" on page 160) and then enclose in *SimpleControl* (on page 196) definitions: Did not meet criteria for randomization, Adverse experience (other than adverse laboratory experience), Adverse laboratory experience, and Lost to follow-up.
- The Other (specify) item is a simple text box with a caption.
- Several items combine captioned text boxes with an additional caption: Patient decision, Physician decision, and Sponsor request. For these items, define the caption that appears next to a radio button as a *PFElement*, and enclose it in a *SimpleControl*. Define the text box and its caption as a *TextControl* (on page 216). Then, create a *GroupControl* (on page 134) that combines the two components.
- The Death item consists of the caption that appears next to a radio button and a group of three pull-down lists that define the date of death. For this item, define the caption that appears next to the radio button as a *PFElement*, and enclose it in a *SimpleControl*. Define the three date components as a *DateTimeControl* (on page 105), and enclose the *DateTimeControl* in a *GroupControl*. Finally, define a *GroupControl* that includes the *SimpleControl* for the caption and the *GroupControl* for the date.

Definition steps:

- 1 Define the *PFElements* (see "*PFElement*" on page 160) you need, and enclose them in *SimpleControl* (on page 196) definitions.
- 2


```

<PFELEMENT REFNAME="CRITERIA" LABEL="Did not meet criteria"
TYPE="STRING" VALUE="CriteriaElement"/>
<PFELEMENT REFNAME="AE" LABEL="Adverse Experience (other than adverse
laboratory experience)" TYPE="STRING" VALUE="AEElement"/>
<PFELEMENT REFNAME="ALE" LABEL="Adverse laboratory experience"
TYPE="STRING" VALUE="ALEElement"/>
<PFELEMENT REFNAME="PD" LABEL="Patient decision" TYPE="STRING"
VALUE="PDElement"/>
<PFELEMENT REFNAME="PHD" LABEL="Physician decision" TYPE="STRING"
VALUE="PHDElement"/>
<PFELEMENT REFNAME="SD" LABEL="Sponsor decision" TYPE="STRING"
VALUE="SDElement"/>
<PFELEMENT REFNAME="DEATHSC" LABEL="Death (Sponsor must be notified and
Serious Adverse Experience Report completed.)" TYPE="STRING"
VALUE="DeathElement"/>
<PFELEMENT REFNAME="LOST" LABEL="Lost to follow-up" TYPE="STRING"
VALUE="LostElement"/>
<PFELEMENT REFNAME="OTHER" LABEL="Other (specify):" TYPE="STRING"
VALUE="OtherElement"/> <SIMPLECONTROL REFNAME="CRITERIA"
NAME="CriteriaElement">
  <ELEMENTREF REFNAME="CRITERIA"/>
</SIMPLECONTROL>
<SIMPLECONTROL REFNAME="AE" NAME="AEElement">
  <ELEMENTREF REFNAME="AE"/>
</SIMPLECONTROL>
<SIMPLECONTROL REFNAME="ALE" NAME="ALEElement">
  <ELEMENTREF REFNAME="ALE"/>
</SIMPLECONTROL>
<SIMPLECONTROL REFNAME="PD" NAME="PDElement">
  <ELEMENTREF REFNAME="PD"/>
</SIMPLECONTROL>
<SIMPLECONTROL REFNAME="PHD" NAME="PHDElement">
  <ELEMENTREF REFNAME="PHD"/>
</SIMPLECONTROL>
<SIMPLECONTROL REFNAME="SD" NAME="AEElement">
  <ELEMENTREF REFNAME="SD"/>
</SIMPLECONTROL>
<SIMPLECONTROL REFNAME="DEATHSC" NAME="DeathElement">
  <ELEMENTREF REFNAME="DEATHSC"/>
</SIMPLECONTROL>
<SIMPLECONTROL REFNAME="LOST" NAME="LostElement">
  <ELEMENTREF REFNAME="LOST"/>
</SIMPLECONTROL>
<SIMPLECONTROL REFNAME="OTHER" NAME="OtherElement">
  <ELEMENTREF REFNAME="OTHER"/>
</SIMPLECONTROL>

```


- ```

 <CONTROLREF REFNAME="REASONTEXT" ORDER="2"/>
 </GROUPCONTROL>
 <GROUPCONTROL REFNAME="LOSTGROUP" UUID="PF_SC_LOST"
 NAME="DEATHGROUP" LAYOUT="HORIZONTAL">
 <CONTROLREF REFNAME="LOST" ORDER="1"/>
 </GROUPCONTROL>

```
- 7 Define a *DateTimeControl* (on page 105) for the Death item.
  - 8 <DATETIMECONTROL REFNAME="COMMONDATE" NAME="COMMONDATE"
 STARTYEAR="1997" ENDYEAR="2002"
 DISPLAYDAY="true" DISPLAYMONTH="true"
 DISPLAYYEAR="true" REQUIREMONTH="true"
 REQUIREDAY="true" REQUIREYEAR="true"/>
  - 9 Define a *GroupControl* (on page 134) that includes the DEATHDATE GroupControl and the *SimpleControl* (on page 196) that defines the Death caption.
  - 10 <GROUPCONTROL REFNAME="DEATHGROUP" UUID="PF\_SC\_DEATH"
 NAME="DEATHGROUP" LAYOUT="HORIZONTAL">
 <CONTROLREF REFNAME="DEATHSC" ORDER="1"/>
 <CONTROLREF REFNAME="COMMONDATE" ORDER="2"/>
 </GROUPCONTROL>
  - 11 Define a *RadioControl* (on page 163) that includes a *Controlref* (on page 56) representing each simple or compound control in the list.
  - 12 <RADIOCONTROL REFNAME="REASONRADIO"
 NAME="REASONRADIO" LAYOUT="VERTICAL"
 UUID="PF\_SC\_REASONCTL">
 <CONTROLREF REFNAME="CRITERIAGROUP" ORDER="1"/>
 <CONTROLREF REFNAME="AEGROUP" ORDER="2"/>
 <CONTROLREF REFNAME="ALEGROUP" ORDER="3"/>
 <CONTROLREF REFNAME="PDECGROUP" ORDER="4"/>
 <CONTROLREF REFNAME="PHDECGROUP" ORDER="5"/>
 <CONTROLREF REFNAME="SDECGROUP" ORDER="6"/>
 <CONTROLREF REFNAME="DEATHGROUP" ORDER="7"/>
 <CONTROLREF REFNAME="LOSTGROUP" ORDER="8"/>
 <CONTROLREF REFNAME="OTHERGROUP" ORDER="9"/>
 </RADIOCONTROL>

## Report

### Purpose

The Report tag enables you to add a report definition to the database.

### Syntax

```
<REPORT
 [UUID="id"]
 ASPFILENAME="name"
 ASPTEXT="code"
 [QUERYFILENAME="name"]
 [QUERYTEXT="text"]
 [TITLE="name"]
 [DESCRIPTION="text"]
 [REPORTTYPE="ADMIN|NORMAL"]
 [DATASOURCENAME="text"]
 [DATASOURCEUSER="text"]
 [DATASOURCEPASSWORD="text"]
 [LANGUAGE="name"]/>
```

### Attributes

**UUID="id" Universally Unique Identifier** (see "**MedML Schema**" on page 11); a string that identifies the component uniquely across all trials, trial databases, and machines. Optional; do not specify a UUID when defining a custom report.

#### **ASPFILENAME="name"**

File name of the ASP page associated with the report. Either ASPFILENAME or ASPTEXT is required.

#### **ASPTEXT="text"**

The actual text of the ASP page associated with the report. Either ASPFILENAME or ASPTEXT is required.

#### **QUERYFILENAME="name"**

The name of a file containing the SQL query that is executed when the report is run. Optional; this attribute is used for documentation only. This attribute cannot be used if the following attribute, QUERYTEXT, is being used.

#### **QUERYTEXT="text"**

The actual text of the SQL query that is executed when the report is run. Optional; this attribute is used for documentation only. This attribute cannot be used if the previous attribute, QUERYFILENAME, is being used.

#### **TITLE="name"**

Title of the report. Optional.

**DESCRIPTION="text"**

Description of the report, for documentation only. Optional.

**REPORTTYPE="ADMIN|NORMAL"**

Type of report. Optional; NORMAL is the default:

- ADMIN—The report is an Admin report and appears under the Admin tab of the Reports module.
- NORMAL—The report is either a predefined or custom report.

**DATASOURCENAME="text"** Name of the ODBC data source name of the database used for reports. Optional; the DSN of the InForm database is the default.

**DATASOURCEUSER="text"**

User Name used to access the data source. Optional; the user name for the InForm database DSN is the default.

**DATASOURCEPASSWORD="text"**

Password used to access the data source. Optional; the password for the InForm database DSN is the default.

**LANGUAGE="name"**

Language used to display the caption. Optional; English is the default but any language can be specified.

## Example

The following example illustrates the Report tags for several predefined reports in the InForm system. <RESOURCEDATA>

```
<REPORT
 UUID="PF_RPT_QUERY_PERF_CRA"
 TITLE="Query Performance by CRA"
 DESCRIPTION="PF_RPT_QUERY_PERF_CRA"
 QUERYFILENAME="craqueryperf.sql"
 ASPFILENAME="craqueryperf.asp"/>
<REPORT
 UUID="PF_RPT_QUERY_STATUS_CRA"
 TITLE="Query Status by CRA"
 DESCRIPTION="PF_RPT_QUERY_STATUS_CRA"
 QUERYFILENAME="craquerystatus.sql"
 ASPFILENAME="craquerystatus.asp"/>
<REPORT
 UUID="PF_RPT_SDV_PERF_CRA"
 TITLE="SV Performance by CRA"
 DESCRIPTION="PF_RPT_SDV_PERF_CRA"
 QUERYFILENAME="crasdvperf.sql"
 ASPFILENAME="crasdvperf.asp"/>
</RESOURCEDATA>
```

## ReportingGroup

### Purpose

Defines the reporting functionality and type of access available to users with reporting rights. Some Reporting groups allow members to access only standard reports; others allow members access to ad hoc reporting.

Note that the only way to remove a user from a group is through the Admin function of the InForm application. You cannot remove a user by running the MedML Installer utility.

### Syntax

```
<REPORTINGGROUP
 GROUPNAME="name"
 GROUPDESCRIPTION="name">
 <USERREF USERNAME="name" />
</REPORTINGGROUP>
```

#### Attributes

**GROUPNAME="name"**

Name of the ReportingGroup Required.

**GROUPDESCRIPTION="name"**

Text describing the ReportingGroup Optional.

#### Children

A ReportingGroup definition can include zero or more *Userref* (on page 232) definitions. Each Userref refers to a previously created *User* (on page 227) definition that identifies one InForm software user.

### Example

The example below shows the syntax for creating a reporting group named "Ad Hoc Users".

```
<REPORTINGGROUP
 GROUPNAME="Ad Hoc Users"
 GROUPDESCRIPTION="Ad Hoc Users">
 <USERREF USERNAME="cra" />
</REPORTINGGROUP>
```

## Resource

### Purpose

The Resource tag identifies a single resource to load.

### Syntax

```
<RESOURCE
 [UUID="id"]
 [FILENAME="file"]
 [DESCRIPTION="text"]
 DATATYPE="TEXT|GIF|JPEG"
 [LANGUAGE="name"]
 [TEXT="text"]
 [DATA="data"]/>
```

### Attributes

**UUID="id" Universally Unique Identifier** (see "MedML Schema" on page 11); a string that identifies the component uniquely across all trials, trial databases, and machines. Either a UUID or a Filename attribute is required unless you include explicit text or a data stream in the TEXT or DATA attribute.

**FILENAME="file"**

Name of the file in which the resource is enclosed. Either a UUID or a Filename attribute is required.

**DESCRIPTION="text"**

Description of the resource. Optional.

**DATATYPE="TEXT|GIF|JPEG"**

Type of data contained in the file: TEXT, GIF, or JPEG. Required.

**LANGUAGE="name"**

Language used in the file. Optional; English is the default.

**Note:** Only English is supported in this release of the InForm application. For trials that run with the InForm application, Oracle recommends leaving the Language attribute out of Resource definitions and letting the MedML Installer utility default to English.

**TEXT="text"**

Actual text of a resource for which the DATATYPE is TEXT. If you include actual text, do not specify a UUID or FILENAME attribute.

**DATA="data"**

Data stream that defines a resource for which the DATATYPE is GIF or JPEG. When the DATATYPE is GIF or JPEG, a UUID, a FILENAME, or a DATA definition is required. If you include a data stream, do not specify a UUID or FILENAME attribute.

## Examples

### Example 1

The following example illustrates two text resources.

```
<RESOURCE FILENAME="PAGE.HTML"
 UUID="b69ff18f=e466-11d1-9e5c-00a0c9769a33"
 DESCRIPTION="PAGE.HTML"
 DATATYPE="TEXT"/>
```

```
<RESOURCE FILENAME="FORM.HTML"
 UUID="b69ff190=e466-11d1-9e5c-00a0c9769a33"
 DESCRIPTION="FORM.HTML"
 DATATYPE="TEXT"/>
```

### Example 2

The following example illustrates three graphical resources.

```
<RESOURCE FILENAME="SideArrow.GIF"
 UUID="56fc2652-ee9f-11d1-a744-00a0c9af7673"
 DESCRIPTION="SideArrow.GIF"
 DATATYPE="GIF"/>
```

```
<RESOURCE FILENAME="SideDocsGray.GIF"
 UUID="573c85d0-ee9f-11d1-a744-00a0c9af7673"
 DESCRIPTION="SideDocsGray.GIF"
 DATATYPE="GIF"/>
```

```
<RESOURCE FILENAME="SideDocsYellow.GIF"
 UUID="577a82f4-ee9f-11d1-a744-00a0c9af7673"
 DESCRIPTION="SideDocsYellow.GIF"
 DATATYPE="GIF"/>
```

## Right

### Purpose

The Right tag enables you to load definitions of rights that identify specific InForm activities that users can perform. The InForm application is delivered with predefined rights.

### Syntax

```
<RIGHT
 RIGHT="name"/>
```

### Attributes

**RIGHT="name"/>** Name of the right.

### Example

The following example illustrates the definitions of some of the rights that are predefined for the InForm application.

```
<RIGHT RIGHT="Create User Right"/>
<RIGHT RIGHT="Modify User Rights"/>
<RIGHT RIGHT="Activate Site User"/>
<RIGHT RIGHT="Deactivate Site User"/>
<RIGHT RIGHT="Activate Sponsor User"/>
<RIGHT RIGHT="DeActivate Sponsor User"/>
<RIGHT RIGHT="Make user active"/>
<RIGHT RIGHT="Modify System Configuration"/>
```

## Rightref

### Purpose

The Rightref tag enables you to include previously defined *Rights* (see "*Right*" on page 175) in the definition of a *RightsGroup* (on page 176). A Rightref appears only as the child of a RightsGroup in which it is included; it is not submitted as a stand-alone component.

### Syntax

```
<RIGHTREF
 RIGHT="name"/>
```

### Attributes

**RIGHT="name"/>** Name of the right referenced by the Rightref. This is the same name as specified in the Right attribute of the *Right* (on page 175) definition.

## Example

The following definition of the PF Admin Rights Group, which identifies the activities that a trial administrator can perform, illustrates how to use `Rightrefs` to include *Right* (on page 175) definitions in a *RightsGroup* (on page 176).

```
<RIGHTSGROUP GROUPNAME="PF Admin Rights Group">
 <RIGHTREF RIGHT="Activate Site User"/>
 <RIGHTREF RIGHT="Deactivate Site User"/>
 <RIGHTREF RIGHT="Activate Sponsor User"/>
 <RIGHTREF RIGHT="DeActivate Sponsor User"/>
 <RIGHTREF RIGHT="Make user active"/>
 <RIGHTREF RIGHT="Create Sites"/>
 <USERREF USERNAME="Krusty"/>
</RIGHTSGROUP>
```

## RightsGroup

### Purpose

The `RightsGroup` tag enables you to create a set of *Rights* (see "*Right*" on page 175) and to assign the set of rights to *Users* (see "*User*" on page 227). The InForm application is delivered with a set of predefined Rights and RightsGroups.

### Syntax

```
<RIGHTSGROUP
 GROUPNAME="name"
 [GROUPDESCRIPTION="text"]
 [UUID="id"]>
 <RIGHTREF* attributes/>
 <USERREF* attributes/>
 <ITEMGROUPREF* attributes/>
</RIGHTSGROUP/>
```

### Attributes

**GROUPNAME="name"** Name of the RightsGroup. Required.

**GROUPDESCRIPTION="text"**

Text describing the RightsGroup. Optional.

**UUID="id"**

*Universally Unique Identifier* (see "*MedML Schema*" on page 11); a string that identifies the component uniquely across all trials, trial databases, and machines. Optional.



## Children

A RightsGroup definition can include zero or more:

- **Rightref**(on page 175) definitions. Each Rightref refers to a previously created **Right** (on page 175) definition that identifies an InForm activity that users can perform.
- **Userref**(on page 232) definitions. Each Userref refers to a previously created **User** (on page 227) definition that identifies one InForm user.
- **ItemGroupref**(on page 145) definitions. Each ItemGroupref refers to a previously created **ItemGroup** (on page 144) definition that identifies a group of **Items** (see "**Item**" on page 139) for which a user can set a rights group- specific display override.

## Example

The following example illustrates the definition of the PF Admin Rights Group, which identifies the activities that a trial administrator can perform. This definition also assigns the administrator rights set to the user named admin.

```
<RIGHTSGROUP GROUPNAME="PF Admin Rights Group">
 <RIGHTREF RIGHT="Activate Site User"/>
 <RIGHTREF RIGHT="Deactivate Site User"/>
 <RIGHTREF RIGHT="Activate Sponsor User"/>
 <RIGHTREF RIGHT="DeActivate Sponsor User"/>
 <RIGHTREF RIGHT="Make user active"/>
 <RIGHTREF RIGHT="Create Sites"/>
 <USERREF USERNAME="admin"/>
</RIGHTSGROUP>
```

In the following example, the CRC RG rights group includes the definitions of the crc **User** (on page 227) and the CRC\_Hidden **ItemGroupref**(on page 145).

```
<RIGHTSGROUP GROUPNAME="CRC RG">
 <RIGHTREF RIGHT="Print" />
 <RIGHTREF RIGHT="Reports" />
 <RIGHTREF RIGHT="Enroll Patients" />
 <RIGHTREF RIGHT="View CRF" />
 <RIGHTREF RIGHT="Enter Data into a CRF" />
 <RIGHTREF RIGHT="Edit Data on a CRF" />
 <RIGHTREF RIGHT="Enter Comments into a CRF" />
 <RIGHTREF RIGHT="Mark a CRF as Ready for SV" />
 <RIGHTREF RIGHT="Mark and unmark a CRB as Ready for SV" />
 <RIGHTREF RIGHT="Answer Query" />
 <RIGHTREF RIGHT="View Connections" />
 <RIGHTREF RIGHT="View Default Connection" />
 <RIGHTREF RIGHT="Synchronize New Data" />
 <USERREF USERNAME="crc" />
 <ITEMGROUPEF REFNAME="CRC_Hidden" DISPLAYOVERRIDE="HIDDEN"/>
</RIGHTSGROUP>
```

## Rule

### Purpose

The Rule tag enables you to define a rule and the script that executes when it fires, and to associate the rule with an *Event* (on page 119).

### Syntax

```
<RULE
 REFNAME="name"
 [DESIGNNOTE="text"]
 [UUID="id"]
 [DESCRIPTION="text"]
 [ENABLED="true | false"]
 [HELPTEXT="text"]

 SCRIPTTYPE="SERVERRULE | BROWSERRULE | SERVERCALCULATION | SERVERCONVERSION |
 SERVERRANDOMIZATION | CLINTRIALDERIVATION | CLINTRIALRULE"
 [SCRIPTTEXT="text"]
 [SCRIPTFILE="file"]
 [MSGISDERIVED="true | false"]
 [EMPTYISTRUE="true | false"]
 [MSGTEXT="text"]
 [RULEACTION="REPORT | REJECT"]>
<RULEARG* attributes/>
<EVENTREF attributes/>
<TESTCASE attributes/>

</RULE>
```

### Attributes

**REFNAME="name"** Name used when attaching a Rule to an *Item* (on page 139) in the *AttachRuleSet* (on page 88) tag. Can be up to 255 characters in length, including spaces. Required.

**DESIGNNOTE="text"**

Free-form text, with a maximum of 255 characters, containing any information you want to capture about the design of the component. This information is for documentation only and is not displayed. Optional.

**UUID="id"**

*Universally Unique Identifier* (see "*MedML Schema*" on page 11); a string that identifies the component uniquely across all trials, trial databases, and machines. Optional.

**DESCRIPTION="text"**

Description of the rule. Optional.

**ENABLED="true | false"**

Indicates whether the Rule is active and will fire under the appropriate circumstances if the Rule is attached to an Item: true or false. Optional; true is the default.

**HELPTEXT="text"**

Description of the data check performed by the rule. This description appears in the CRF help for the data item to which the rule is attached, if the CRF help includes the appropriate mapping. You can create context-specific help that varies with the Items or ItemSets to which a rule is attached. The InForm application substitutes values defined as rule arguments for each Item or Itemset into the help text. This enables you to use the same help text for multiple applications of the rule. Optional.

**SCRIPTTYPE="SERVERRULE | BROWSERRULE | SERVERCALCULATION | SERVERCONVERSION | SERVERRANDOMIZATION | CLINTRIALDERIVATION | CLINTRIALRULE"**

Purpose of the Rule's script; required:

- **Serverrule**—The rule executes on the server.
- **Browserrule**—The rule executes on the browser. Two browser-side rule activities are supported in the InForm application: checks for text entries in a numeric field or numeric entries in an alphabetic field, and smart controls. Smart controls provide automatic selection of radio buttons and check boxes when a user enters or selects a value in an associated text control or list.
- **Servercalculation**—The rule provides the result for a calculated data item whose value is based on the value of one or more other items.
- **Serverconversion**—The rule converts units from one standard to another.
- **Serverrandomization**—The rule determines the *randomized drug kit number* (see "*Randomization objects and methods*" on page 242) to assign in a calculated control.
- **Clintrialderivation**—The rule is used in a Clintrial trial and provides the result for a derived data item whose value is based on the value of one or more other items in the Clintrial database.
- **Clintrialrule**—The rule is used in a Clintrial trial.

**SCRIPTTEXT="text"** Text of the script. Only Visual Basic scripts are supported. Either the Scripttext or Scriptfile attribute is required.

**SCRIPTFILE="file">**

Name of the file that contains the text of the script. Only Visual Basic scripts are supported. Either the Scripttext or Scriptfile attribute is required.

**MSGISDERIVED="true | false"**

In a Clintrial rule, indicates whether the rule text is generated at runtime by a Clintrial derivation: true or false. Required if the rule is a Clintrial rule. If true, do not specify a MSGTEXT value.

**EMPTYISTRUE="true | false"**

In a Clintrial rule, indicates whether a null return value translates to true or false. Required if the rule is a Clintrial rule.

**MSGTEXT="text"**

Text displayed when a Clintrial rule fails. Optional; if present, must be 240 characters or less. Do not specify if the rule is a Clintrial rule.

**RULEACTION="REPORT | REJECT"**

In a Clintrial rule, indicates whether the validation status of a failed rule can be overridden; required if the rule is a Clintrial rule:

- Report—The validation status can be set to Passed Validation (0) even if the rule fails.
- Reject—The record must have a Failed Validation status (-1).

**Children**

A Rule definition can include the following components:

- Zero or more **Rulearg** (on page 182) definitions are allowed. Rulearg definitions contain parameters that are passed into a rule script.
- One **Event** (on page 119) definition is required to define the action that results if a rule fails. To associate an Event with a Rule, use the **Eventref** (on page 120) tag.
- Zero or more **Testcase** (on page 214) definitions are allowed. Testcase definitions contain test data and expected results that you can run against a rule with the InForm Architect application.

**Example**

The following Rule definition includes the evtGeneric **Event** (on page 119) definition with an **Eventref** (on page 120) tag. Additionally, **Rulearg** (on page 182) tags are used to define the arguments min, max, qtext, and addpath. Four values are supplied as test cases with the **Testcase** (on page 214) tag.

```

<RULE REFNAME="rulRangeCheck"
 DESCRIPTION="Ensure data are within upper and lower range"
 ENABLED="true"
 SCRIPTTYPE="SEVERRULE">
<![CDATA[Name : rulRangeCheck
'Desc :Compares minimum and maximum ranges entered to ensure that
'data are within the expected range. This is a generic rule that can
'be attached to any item that has an upper and lower range.
'Args:
'min - minimum value
'max - maximum value
'qtext - querytext
'addpath - additional path to the control (beyond the item)
'

```

---

Option Explicit

Dim Min, Max, qtext, addpath, item, v\_val

Min = Patient.GetArgument("min")

Max = Patient.GetArgument("max")

qtext = Patient.GetArgument("qtext")

addpath = Patient.GetArgument("addpath")

item = Patient.GetCurPath() & addpath

Patient.AddNamedValue "QUERYTEXT", qtext

Result.Rulepassed = 1>true

v\_val = Patient.GetValue(item, "", 0,0,0)

If cDbl(v\_val) < cDbl(Min) or cDbl(v\_val) > cDbl(Max) then

    Result.Rulepassed = 0

End If

]]>

<EVENTREF REFNAME="evtGeneric"/>

<RULEARG

    NAME="addpath"

    TYPE="STRING"/>

<RULEARG

    NAME="qtext"

    TYPE="STRING"/>

<RULEARG

    NAME="min"

    TYPE="STRING"/>

<RULEARG

    NAME="max"

    TYPE="STRING"/>

<TESTCASE

    RESULT="Pass"

    INPUT\_1="15"

    INPUT\_2="22"

    INPUT\_3="93"

    INPUT\_4="2"/>

</RULE>

## Rulearg

### Purpose

The RULEARG tag enables you to pass an argument in to a parameter referenced by the GetArgument method in a rule script. By creating sets of RULEARG definitions and associating them with different Items or ItemSets in multiple ATTACHRULESET definitions, you can use the same rule script over and over to test various value ranges in multiple Items or ItemSets. Additionally, you can use the RULEARG tag to customize the CRF Help description of the data check performed by a rule.

### Syntax

```
<RULEARG
 NAME="name"
 VALUE="value"
 TYPE="STRING|NUMERIC|FLOAT|DATE"/>
```

### Attributes

**NAME="name"** The name of the Rule for which you are specifying values. Required.

**VALUE="value"**

The values you want to plug into the Rule. Required.

**TYPE="STRING|NUMERIC|FLOAT|DATE"**

*String, Integer, Float or Date. Required;String is the default.*

## Example

The following example attaches the Height Range Checking Rule to the Height item on the Demographics form of Visit 1. The rule looks at the value in this item and compares it to the parameters and the values specified for those parameters in the RuleArg in AttachRuleSet. If the InForm application does not find RuleArg values in AttachRuleSet, it looks for default values in the Rule. In the example below, the minimum value for height is 34 as specified in AttachRuleSet and since the maximum value is not defined in AttachRuleSet, InForm software uses the maximum value defined in the Rule, which is 90.

If there are no RULEARG values, it looks for default values in the Systolic Range Checking Rules.

```
<RULE REFNAME="Height Range Checking Rule"
 DESCRIPTION="Height Range Checking Rule"
 ENABLED="true"
 SCRIPTTYPE="SERVERRULE"
 SCRIPTFILE="HeightRangeCheck.vbs">
<RULEARG NAME="Min" TYPE="NUMERIC"/>
<RULEARG NAME="Max" VALUE="90" TYPE="NUMERIC"/>
<EVENTREF REFNAME="Height Range"/>
</RULE>

<ATTACHRULESET REFNAME="Height Range Checking Rule" RULENAME="Height Range
Checking Rule"
 FORMSETNAME="Visit1"
 FORMNAME="DEM"
 SECTIONNAME="DEM"
 ITEMNAME="HEIGHT"
 HELPTEXT="Height is expected to be between 34- 76 IN or 88-191 CM."
 ATTACHTYPE="true"
 ACTIVE="true">
<RULEARG NAME="Min" VALUE="34" TYPE="NUMERIC"/>
</ATTACHRULESET>
```

## Section

### Purpose

The Section tag defines a grouping, under a section heading, of related *Items* (see "*Item*" on page 139) or one *ItemSet* (on page 149) on a form.

### Syntax

```
<SECTION
 REFNAME="name"
 [DESIGNNOTE="text"]
 [TITLE="text"]
 [UUID="id"]
 [NOTE="text"]
 [LANGUAGE="name"]
 [REPEATING="true | false"]>
 <ITEMREF+ attributes/>
</SECTION>
```

### Attributes

**REFNAME="name"** Name used when referring to the Section in the definition of a Form. Required. This name must be unique among Sections and must not exceed 31 characters.

### DESIGNNOTE="text"

Free-form text, with a maximum of 255 characters, containing any information you want to capture about the design of the component. This information is for documentation only and is not displayed. Optional.

### TITLE="text"

Text of the Section title, displayed at the top of the Section on the screen. Optional.

### UUID="id"

*Universally Unique Identifier* (see "*MedML Schema*" on page 11); a string that identifies the component uniquely across all trials, trial databases, and machines. Optional.

**Note:** When you create a repeating *FormSet* (on page 127), often to identify an unscheduled visit, the first form in the FormSet must contain a Section definition that includes the following UUID: **BD991BBE-B0A4-11D2-80E3-00A0C9AF7674**

### NOTE="text"

Text of a note displayed immediately below the Section title. Optional.

### LANGUAGE="name"

Language used to display the Section title and note. Optional; English is the default.



**REPEATING="true | false"**

Indicates whether the section contains a repeating *ItemSet* (on page 149) definition: true or false. Optional; false is the default. Note the following rules about repeating ItemSets:

- You can add an ItemSet component only to a Section definition with the REPEATING attribute set to true.
- Only one ItemSet definition can appear in a REPEATING Section definition.
- Regular, nonrepeating *Item* (on page 139) definitions cannot appear in a REPEATING Section definition.

**Children**

A Section definition must include one or more *Itemref* (on page 72) definitions. Each Itemref refers to a previously defined *Item* (on page 139) or *ItemSet* (on page 149).

**Note:** When you create a repeating *FormSet* (on page 127), often to identify an unscheduled visit, the first form in the FormSet must contain a Date of Visit Section definition consisting of one *Item* (on page 139)—a *DateTimeControl* (on page 105) definition that includes the month, day, year, hour, and minute of the visit. The Section, Item, and DateTimeControl definitions must include specific *UUID* (see "*MedML Schema*" on page 11)s, as noted in the Example section. Note that the InForm Architect application and the MedML Installer utility convert alphabetic characters in UUIDs to uppercase.

**Examples****Example 1**

This example illustrates the creation of a Section called Duration of Hypertension. The Section is composed of two *Itemrefs* (see "*Itemref*" on page 72):

- The first Itemref refers to an *Item* (on page 139) that consists of a question and a *RadioControl* (on page 163). For the purpose of this example, assume that a Yes/No RadioControl has been defined previously.
- The second Itemref refers to an Item that consists of a question and a *GroupControl* (on page 134). The GroupControl is composed of two *TextControls* (see "*TextControl*" on page 216), each of which includes a *Unit* (on page 220) definition by using a *Unitref* (on page 223) tag.

Duration of Hypertension		
16.	Was hypertension previously diagnosed:	<input type="radio"/> Yes <input type="radio"/> No
17.	If hypertension was previously diagnosed, enter duration of hypertension:	Years <input type="text"/> Months <input type="text"/>

- 1 Create *Unit* (on page 220) definitions for years and months.

- 2 

```
<UNIT REFNAME="YEARS" SYMBOL="Year(s)" CLASSIFICATION="Time"
 BASEREFNAME="MONTHS" CONVERSIONTOBASE="12"
 CONVERSIONFROMBASE=".0833
 UUID="498100f6-e9df-11d1-9e60- 00a0c9769a33"/>
<UNIT REFNAME="MONTHS" SYMBOL="Month(s)" CLASSIFICATION="Time"
 BASEREFNAME="MONTHS" CONVERSIONTOBASE="1"
 CONVERSIONFROMBASE="1"
 UUID="498100f7-e9df-11d1-9e60- 00a0c9769a33"/>
```
- 3 Enclose the years and months Units in *TextControl* (on page 216) definitions by using *Unitref* (on page 223) tags.
- 4 

```
<TEXTCONTROL REFNAME="HTYEARSTXT"
 NAME="HTYEARSTXT"
 HEIGHT="1"
 LENGTH="3"
 MAXLENGTH="3"
 DATATYPE="INTEGER">
 <UNITREF* attributes/>
</TEXTCONTROL> <TEXTCONTROL REFNAME="HTMONTHSTXT"
 NAME="HTMONTHSTXT"
 HEIGHT="1"
 LENGTH="3"
 MAXLENGTH="3"
 DATATYPE="INTEGER">
 <UNITREF* attributes/>
</TEXTCONTROL>
```
- 5 Create a *GroupControl* (on page 134) definitions by using the TextControls.
- 6 

```
<GROUPCONTROL REFNAME="HTDURATIONGRP"
 NAME="HTDURATIONGROUP"
 LAYOUT="HORIZONTAL">
 <CONTROLREF REFNAME="HTYEARSTXT" ORDER="1"/>
 <CONTROLREF REFNAME="HTMONTHSTXT" ORDER="2"/>
</GROUPCONTROL>
```
- 7 Create *Item* (on page 139) definitions for the two line items in the Section.
- 8 

```
<ITEM REFNAME="HTYESNO" QUESTION="Was hypertension previously diagnosed?
">
 <CONTROLREF REFNAME="YESNORADIO"/>
</ITEM>
<ITEM REFNAME="HTDURATION" QUESTION="If hypertension was previously
diagnosed,
 enter duration of hypertension: ">
 <CONTROLREF REFNAME="HTDURATIONGRP"/>
</ITEM>
```
- 9 Use *Itemrefs* (see "*Itemref*" on page 72) to enclose the Item definitions in the definition of the Duration of Hypertension section.

```

10 <SECTION REFNAME="HYPERTENSION"
 TITLE="Duration of Hypertension">
 <ITEMREF REFNAME="HTYESNO" ORDER="1"/>
 <ITEMREF REFNAME="HTDURATION" ORDER="2"/>
</SECTION>

```

## Example 2

This example illustrates the Date of Visit Section definition that is required in the first form of every visit and on the first form of a repeating *FormSet* (on page 127). The section must contain only one *Item* (on page 139)—the DOV item, which enables users to enter the date and time of the visit. We recommend that you create this section as a separate *Form* (on page 122).

Date Of Visit	
1.	Date and time of visit: <input type="text"/> / <input type="text"/> / <input type="text"/> <input type="text"/> : <input type="text"/>

```

<DATETIMECONTROL REFNAME="DOV" NAME="DOV"
 UUID="BD991BC0-B0A4-11D2-80E3- 00A0C9AF7674"
 STARTYEAR="1997" ENDYEAR="2004"
 DISPLAYDAY="true" DISPLAYMONTH="true"
 DISPLAYYEAR="true" DISPLAYHOUR="true"
 DISPLAYMINUTE="true"
 REQUIREMONTH="true" REQUIREYEAR="true"/>

```

```

<ITEM REFNAME="DOV" QUESTION="Date and time of visit:"
 UUID="BD991BBF-B0A4-11D2-80E3-00A0C9AF7674">
 <CONTROLREF REFNAME="DOV"/>
</ITEM>

```

```

<SECTION REFNAME="DOV" TITLE="Date Of Visit"
 UUID="BD991BBE-B0A4-11D2-80E3-00A0C9AF7674">
 <ITEMREF REFNAME="DOV" ORDER="1"/>
</SECTION>

```

```

<FORM REFNAME="DOV" TITLE="Date of Visit" MNEMONIC="DOV" TYPE="CRF">
 <SECTIONREF REFNAME="DOV"/>
</FORM>

```

## Sectionref

### Purpose

The Sectionref tag enables you to:

- Include previously defined *Sections* (see "*Section*" on page 184) in the definition of a *Form* (on page 122). Include one Sectionref tag for each Section in the Form.
- Specify the RefName of a Section as the location of a mapped control in the *Path* (on page 82) tag of a mapping definition. Include one Sectionref tag in a RefName path defined by a Path tag

A Sectionref appears only as the child of a Form or Path tag in which it is included; it is not submitted as a stand-alone component.

### Syntax

```
<SECTIONREF
 REFNAME="name"
 [ORDER="n"]/>
```

### Attributes

**REFNAME="name"** RefName of the *Section* (on page 184) that the Sectionref is including in the definition of a *Form* (on page 122) or *Path* (on page 82). Required.

**ORDER="n"**

Sequence in which each Sectionref appears in the Form definition. Optional. If you do not specify an order, the MedML Installer utility orders the Items referred to by the Sectionrefs in the order in which you enter them.

**Note:** When including a Sectionref definition in a *Path* (on page 82) tag, only the REFNAME attribute is valid.

### Example

This example illustrates the use of Sectionrefs in the definition of the Demographics form, which contains two *Sections* (see "*Section*" on page 184), Demographics (DEM) and Smoking History (SH). For more details about the definition of a form, see the *Form* (on page 122) topic.

```
<FORM REFNAME="DEM" TITLE="Demographics" MNEMONIC="DEM"
 FORMTYPE="CRF">
 <SECTIONREF REFNAME="DEM"/>
 <SECTIONREF REFNAME="SH"/>
</FORM>
```

The following example shows the LEADEC section as the location where the mapped COMMONDATE control occurs.

```
<PATH>
 <CHAPTERREF REFNAME="PF_ALL_VISITS"/>
 <PAGEREF REFNAME="ECG"/>
 <SECTIONREF REFNAME="LEADEC"/>
 <ITEMSETREF REFNAME="1"/>
 <ITEMREF REFNAME="DATEASSESS"/>
 <CONTROLREF REFNAME="COMMONDATE"/>
</PATH>
```

## Sequence

### Purpose

The Sequence component specifies a numbering sequence. Sequences enable Sponsors to indicate how numbers are assigned to screened patients, enrolled patients, queries, and other types of numbered data. Each Sequence is associated with a sequence type, defined with the *SequenceType* (on page 191) tag.

### Syntax

```
<SEQUENCE
 UUID="id"
 SEQUENCENAME="name"
 SEQUENCETYPENAME="name" />
```

### Attributes

**UUID="*id*"** *Universally Unique Identifier* (see "*MedML Schema*" on page 11); a string that identifies the component uniquely across all trials, trial databases, and machines. Required. Note that the InForm Architect application and the MedML Installer utility convert alphabetic characters in UUIDs to uppercase.

**SEQUENCENAME="*name*"**

Descriptive name of the type of number for which you are creating a sequence definition. Required.

**SEQUENCETYPENAME="*name*"**

Name of the *SequenceType* (on page 191) to which the sequence number belongs. Required.

## Example

The following example shows the XML tags used to define query, enrollment, screening, and randomization number sequences.

```
<SEQUENCE SEQUENCENAME="Query Number Sequence"
 SEQUENCETYPENAME="Query"
 UUID="5b7d4eb4-0465-11d2-a414-00a0c963e0ac" />
```

```
<SEQUENCE SEQUENCENAME="Enrollment Number Sequence"
 SEQUENCETYPENAME="Enrollment"
 UUID="eb75b898-078b-11d2-a417-00a0c963e0ac" />
```

```
<SEQUENCE SEQUENCENAME="Screening Number Sequence"
 SEQUENCETYPENAME="Screening"
 UUID="f7f1b3b8-0b5c-11d2-a418-00a0c963e0ac" />
```

```
<SEQUENCE SEQUENCENAME="Simple SimpleCentral"
 SEQUENCETYPENAME="Randomization"
 UUID="4F4A0246-5009-11d2-931C- 00A0C9769A13" />
```

## SequenceType

### Purpose

The SequenceType tag defines the types of entities for which the InForm application automatically generates sequential numbers. When you create a definition of the numerical sequence by using the **Sequence** (on page 189) tag, you specify a type to which the sequence belongs by including a SequenceType definition. Currently InForm application defines the following types of sequence numbers:

- Screening
- Enrollment
- Query
- Randomization

### Syntax

```
<SEQUENCETYPE
 [UUID="id"]
 SEQUENCETYPENAME="name" />
```

### Attributes

**UUID="id"** *Universally Unique Identifier* (see "*MedML Schema*" on page 11); a string that identifies the component uniquely across all trials, trial databases, and machines. Optional.

**SEQUENCETYPENAME="name"**

Name of the sequence type. Required.

### Example

The following SequenceType definitions create the base sequence types used in the InForm application:

```
<SEQUENCETYPE SEQUENCETYPENAME="Enrollment"/>
<SEQUENCETYPE SEQUENCETYPENAME="Screening"/>
<SEQUENCETYPE SEQUENCETYPENAME="Query"/>
<SEQUENCETYPE SEQUENCETYPENAME="Randomization"/>
```

## SignatureGroup

### Purpose

A signature group specifies a set of users who can sign Case Books or CRFs.

Note that the only way to remove a user from a group is through the Admin function of the InForm application. You cannot remove a user by running the MedML Installer utility.

### Syntax

```
<SIGNATUREGROUP
 GROUPNAME="name"
 [GROUPDESCRIPTION="text"]
 [UUID="id"]
 [LANGUAGE="name"]
 [CRFTEXT="text"]
 [CRFFILE="file"]
 [CRFMEANING="text"]
 [CRBTEXT="text"]
 [CRBFILE="file"]
 [CRBMEANING="text"]>
 <USERREF* attributes/>
</SIGNATUREGROUP>
```

### Attributes

**GROUPNAME="name"** Name of the SignatureGroup. Required.

**GROUPDESCRIPTION="text"**

Text describing the SignatureGroup. Optional.

**UUID="id"**

*Universally Unique Identifier* (see "*MedML Schema*" on page 11); a string that identifies the component uniquely across all trials, trial databases, and machines. Optional.

**LANGUAGE="name"**

Language used to display the question. Optional; English is the default.

**CRFTEXT="text"**

Text of the Electronic Signature Affidavit displayed to members of the signature group when signing a CRF associated with the signature group. Optional; if used, specify either the CRFTEXT or CRFFILE attribute. If you do not specify either attribute, the MedML Installer utility uses the text provided in a default CRF signing text resource. The text must include:

- What the signature means - for example, attestation to the accuracy of the data provided in the form
- Stated intention of the signer for the electronic signature to be the legal equivalent of a handwritten signature



Optionally, the text can include string substitution characters (%s) to represent the user's first and last names. See the Example section. **CRFFILE="file"**

Path name of an HTML or text file containing the text of the Electronic Signature Affidavit displayed to members of the signature group when signing a CRF associated with the signature group. The path name must be relative to the directory from which you run the MedML Installer utility. Optional; if used, specify either the CRFTEXT or CRFFILE attribute. If you do not specify either attribute, the MedML Installer utility uses the text provided in a default CRF signing text resource.

**CRFMEANING="text"**

Text that summarizes the meaning of the signature on the CRF. This text is displayed on the Signature Details screen and in the list of completed and required signatures on the CRF.

**CRBTEXT="text"**

Text of the Electronic Signature Affidavit displayed to members of the signature group when signing a Case Book associated with the signature group. Optional; if used, specify either the CRBTEXT or CRBFILE attribute. If you do not specify either attribute, the MedML Installer utility uses the text provided in a default Case Book signing text resource. The text must include:

- What the signature means - for example, attestation to the accuracy of the data provided in the form
- Stated intention of the signer for the electronic signature to be the legal equivalent of a handwritten signature

Optionally, the text can include string substitution characters (%s) to represent the user's first and last names. See the Example section. **CRBFILE="file"**

Path name of an HTML or text file containing the text of the Electronic Signature Affidavit displayed to members of the signature group when signing a Case Book associated with the signature group. The path name must be relative to the directory from which you run the MedML Installer utility. Optional; if used, specify either the CRBTEXT or CRBFILE attribute. If you do not specify either attribute, the MedML Installer utility uses the text provided in a default Case Book signing text resource.

**CRBMEANING="text"**

Text that summarizes the meaning of the signature on the Case Book. This text is displayed on the Signature Details screen and in the list of completed and required signatures on the CRF used for signing a Case Book.

## Children

A SignatureGroup definition can include zero or more *Userref* (on page 232) definitions. Each Userref refers to a previously created *User* (on page 227) definition that identifies one InForm application user.

## Example

In the following SignatureGroup, users Apu and George can sign.

```
<SIGNATUREGROUP GROUPNAME="CRA Signature">
 <USERREF USERNAME="Apu"/>
 <USERREF USERNAME="George"/>
</SIGNATUREGROUP>
```

The following SignatureGroup definition illustrates the inclusion of the signing text for a CRF. The two %s characters specify that the displayed Electronic Signature Affidavit should include the user's first and last names. Note that the <font> tag attributes must be in single quotes because they appear within the double quotes of the CRFTEXT attribute.

```
<SIGNATUREGROUP GROUPNAME="PI Signature"
 CRFTEXT="By my dated signature below,
 I, %s %s, verify that this case report form accurately displays
 the results of the examinations, tests, evaluations and
 treatments noted within.

 Pursuant to Section 11.100 of Title 21 of the Code of Federal
 Regulations, this is to certify that I intend that this
 electronic signature is to be the legally binding equivalent
 of my handwritten signature.

 To this I do attest by supplying my user name and password and clicking
 the button marked Submit below.">
 <CRFMEANING="Approval">
 <USERREF USERNAME="Apu"/>
 <USERREF USERNAME="George"/>
</SIGNATUREGROUP>
```

## SignCRF

### Purpose

The SignCRF tag enables you to identify forms that require signature and to specify which signature group a user must belong to in order to be able to sign the form.

### Syntax

```
<SIGNCRF
 SIGNATUREGROUPNAME="name"
 FORMREFNAME="name"
 [RESETFORMSTATE="true | false"]
 [INVALIDATIONLEVEL="USER | GROUP"]
 [FINALCRF="true | false"]/>
```

### Attributes

**SIGNATUREGROUPNAME="name"** RefName of a *SignatureGroup* (on page 192) that contains users who are authorized to sign the form specified in the FORMREFNAME attribute. Required.

**FORMREFNAME="name"**

RefName of a Form that must be signed by a user who is a member of the SignatureGroup specified in the SIGNATUREGROUPNAME attribute. Required.

**Note:** The Screening and Enrollment forms are not intended to be signed. Do not include their RefNames in a SignCRF definition.

**RESETFORMSTATE="true | false"**

Indicates whether associating a new signature group with a form that is fully signed resets the state of the form to not fully signed. When a form is reset, the original signatures remain valid; however, the form must now be signed by a member of the newly-associated signature group as well. Optional; false is the default.

**INVALIDATIONLEVEL="USER | GROUP"**

Specifies whether a signature should be invalidated when a data item is imported after the form has been signed, for example, when a coded value is imported with the Central Coding application. Optional. Values are:

- **USER**—The form or Case Book signature is invalidated if the user who signed can view the item being imported.
- **GROUP**—The form or Case Book signature is invalidated if at least one user in the signature group can view the item being imported.

If you do not specify the INVALIDATIONLEVEL attribute, the InForm application invalidates the signature whenever the form is edited, either directly or by import. **FINALCRF="true | false"**

Indicates whether signing the form specified in the FORMREFNAME attribute signs entire the Case Book. Optional; false is the default.

## Example

The following example illustrates the use of the SignCRF tag to designate the DEM, VS, and SC forms as requiring signature by a member of the CRA Signature signature group. The FINALCRF attribute indicates that signing the SC form signs the case book.

```
<SIGNCRF SIGNATUREGROUPNAME="CRA Signature" FORMREFNAME="DEM"/>
<SIGNCRF SIGNATUREGROUPNAME="CRA Signature" FORMREFNAME="VS"
 INVALIDATIONLEVEL="USER"/>
<SIGNCRF SIGNATUREGROUPNAME="CRA Signature" FORMREFNAME="SC"
 FINALCRF="true"/>
```

## SimpleControl

### Purpose

The SimpleControl tag defines an option in a compound control such as a list of radio buttons or of check boxes. Using SimpleControls enables you to define compound controls in which the individual options are created as different types of controls—for example, a radio button list that has one option the user selects explicitly and one option in which the user enters text. The following types of compound controls can include one or more SimpleControls:

- *CheckBoxControl* (on page 95)
- *GroupControl* (on page 134)
- *Item* (on page 139)
- *RadioControl* (on page 163)

### Syntax

```
<SIMPLECONTROL
 REFNAME="name"
 [DESIGNNOTE="text"]
 [NAME="name"]
 [UUID="id"]
 [CAPTION="text"]
 [LANGUAGE="name"]
 [CAPTIONALIGN="LEFT|RIGHT|TOP|BOTTOM"]
 [ALIGN="LEFT|CENTER|RIGHT|TOP|MIDDLE|BOTTOM">]
 [UNITDISPLAYTYPE="ELEMENT">
 <ELEMENTREF+ attributes/>
 <UNTREF attributes/>
</SIMPLECONTROL>
```

## Attributes

**REFNAME="name"** Name used when referring to the SimpleControl in the definition of another control. Required. This name must be unique among SimpleControls.

**DESIGNNOTE="text"**

Free-form text, with a maximum of 255 characters, containing any information you want to capture about the design of the component. This information is for documentation only and is not displayed. Optional.

**NAME="name"**

Name used when referring to the SimpleControl in the definition of another control. Optional.

**UUID="id"**

*Universally Unique Identifier* (see "*MedML Schema*" on page 11); a string that identifies the component uniquely across all trials, trial databases, and machines. Optional.

**CAPTION="text"**

Text that appears on the screen with the SimpleControl. Optional. This caption provides the opportunity to define additional text to accompany the caption defined for a PFElement.

**LANGUAGE="name"**

Language used to display the caption. Optional; English is the default.

**CAPTIONALIGN="LEFT | RIGHT | TOP | BOTTOM"**

Position of the caption relative to the calculated value: Left, Right, Top, or Bottom. Optional; Left is the default.

**ALIGN="LEFT | CENTER | RIGHT | TOP | MIDDLE | BOTTOM"**

Alignment of the PFElement within the control: Left, Center, Right, Top, Middle, or Bottom. Optional; Left is the default.

**UNITDISPLAYTYPE="ELEMENT"**

Type of control used to display *Units* (see "*Unit*" on page 220) included in the SimpleControl with a *Unitref* (on page 223) definition: Element is the only valid value.

## Children

A SimpleControl serves as a wrapper for one PFElement. Therefore, the SimpleControl definition takes exactly one *Elementref* (on page 117) definition. Additionally, a SimpleControl definition can include one *Unitref* (on page 223) definition, which refers to a previously defined *Unit* (on page 220).

## Example

This example demonstrates how to define a set of PFElements for specifying gender and wrap them in SimpleControls. The SimpleControls are then available for inclusion in a compound control such as a RadioControl:

- 1 Create *PFElements* (see "*PFElement*" on page 160) that define selections for "Male" and "Female."
- 2 

```
<PFELEMENT REFNAME="MALE" LABEL="Male" TYPE="STRING"
VALUE="MElement"/>
<PFELEMENT REFNAME="FEMALE" LABEL="Female" TYPE="STRING"
VALUE="FElement"/>
```
- 3 Create SimpleControls that use an *Elementref* (on page 117) to include the definition of each PFElement.
- 4 

```
<SIMPLECONTROL REFNAME="MALE">
 <ELEMENTREF REFNAME="MALE"/>
</SIMPLECONTROL>
<SIMPLECONTROL REFNAME="FEMALE">
 <ELEMENTREF REFNAME="FEMALE"/>
</SIMPLECONTROL>
```

For an example of how to use SimpleControls in a list of radio buttons, see the *RadioControl* (on page 163) section.

## Site

### Purpose

The Site tag defines a trial location.

### Syntax

```
<SITE
 [NAME="name"]
 [MNEMONIC="name"]
 [ADDRESS="addr1"]
 [ADDRESS2="addr2"]
 [CITY="name"]
 [STATE="name"]
 [PROVINCE="name"]
 [ZIPCODE="code"]
 [POSTCODE="code"]
 [COUNTRY="name"]
 [PHONE="num"]
 [ALTPHONE="num"]
 [FAX="num"]
 [EMAIL="addr"]
 [TIMEZONE="name"]
 STARTDATE="date"
 [ENDDATE="date"]>
```

```
[SITESERVER="server name"]
[BEEPER="beeper number"]
```

```
[SITEDATEFORMAT="MONTH_DAY_YEAR|DAY_MONTH_YEAR|YEAR_MONTH_DAY
Y"]>
</SITE>
```

### Attributes

**NAME="name"** Name of the site. Required.

**MNEMONIC="name"**

Abbreviated name with which to refer to the site. Required.

**ADDRESS="addr1"**

First line of the site address. Optional.

**ADDRESS2="addr2"**

Second line of the site address. Optional.

**CITY="name"**

City in which the site address is located. Optional.

**STATE="name"**

State in which the site address is located. Optional.

**PROVINCE="name"**

Province in which the site address is located. Optional.

**ZIPCODE="code"**

Site Zip code. Optional.

**POSTCODE="code"**

Site postal code. Optional.

**COUNTRY="name"**

Country in which the site address is located. Optional.

**PHONE="num"**

Site telephone number. Optional.

**ALTPHONE="num"**

Site alternate telephone number. Optional.

**FAX="num"**

Site fax number. Optional.

**EMAIL="addr"**

E-mail address used for contacting the site. Optional.

**TIMEZONE="name"**

Time zone in which the site is located, used to convert from internal universal system time to local time. The value for this attribute must be one of the following:

- One of the sub-key names listed in the Windows registry key  
HKEY\_LOCAL\_MACHINE\Software\Microsoft\WindowsNT\CurrentVersion\Time Zones.

This option ensures that the SITE MedML is processed for any operating system locale.

- The Display name of the InForm server operating system locale.

This option allows the SITE MedML to be processed only on an operating system locale that matches the Display name.

Required.

Required.

**STARTDATE="date"**

Date that the site came on line. Required. Users cannot add data for a site before the specified date. Note the following considerations for specifying date information:

- Year values must be between 100 and 9999, inclusively. Always enter the full year, even in abbreviated date formats.
- Many date and time formats are valid. The following table gives examples:

Format	Example
"dd month yyyy"	"25 January 1996"
"hh:mm:ss" (12- hour clock)	"8:30:00"
"hh:mm:ss" (24- hour clock)	"20:30:00"
"month dd, yyyy hh:mm:ss:"	"January 25, 1996 8:30:00"
"hh:mm:ss mon dd, yyyy"	"8:30:00 Jan. 25, 1996"
"mm/dd/yyyy hh:mm:ss"	"1/25/1996 8:30:00"

**ENDDATE="date"**

Date that the site came off line; for example, date that the last patient was signed off and locked. Optional.

**SITESERVER="server name"**

Name of the server designated as the site server. The site server is dedicated for specific activities such as randomization, screening and enrollment, and generating patient numbers.

**SITEDATEFORMAT="MONTH\_DAY\_YEAR"**

The format of the date as you want it to be displayed for the site, if a format isn't specified at the user level. Optional.



## Example

The following example defines a Site for Ace University.

```
<SITE NAME="Ace University"
 MNEMONIC="ACEU"
 ADDRESS="9999 University Avenue"
 ADDRESS2="2nd Floor, Science Hall"
 CITY="Duluth"
 STATE="MN"
 ZIPCODE="55067"
 PHONE="218-555-8888"
 FAX="218-555-9999"
 EMAIL="drjones@aceu.edu"
 TIMEZONE="CDT"
 STARTDATE="10/23/1998">
</SITE>
```

## SiteGroup

### Purpose

A SiteGroup specifies a group of users who have access to a named site.

Note that the only way to remove a user from a group is through the Admin function of the InForm application. You cannot remove a user by running the MedML Installer utility.

### Syntax

```
<SITEGROUP
 SITENAME="name">
 <USERREF* attributes/>
</SITEGROUP>
```

### Attributes

**SITENAME="name"** Name of the site to which the SiteGroup gives access. Required. This name corresponds to the Name attribute in the *Site* (on page 198) definition for the site.

### Children

A SiteGroup definition can include zero or more *Userref* (on page 232) definitions. Each Userref refers to a previously created *User* (on page 227) definition that identifies one InForm software user.

## Example

In the following example, users Marge and Jonah are assigned to a SiteGroup for the Beth Israel site.

```
<SITEGROUP SITENAME="Beth Israel">
 <USERREF USERNAME="Marge"/>
 <USERREF USERNAME="Jonah"/>
</SITEGROUP>
```

## Sponsor

### Purpose

The Sponsor tag defines a trial sponsor. The Sponsor definition is for documentation purposes.

### Syntax

```
<SPONSOR
 [NAME="name"]
 [PROGRAM="text"]
 [THERAPEUTICAREA="text"]
 [NOTE="text"]
 [ADDRESS="addr1"]
 [ADDRESS2="addr2"]
 [CITY="name"]
 [STATE="name"]
 [PROVINCE="name"]
 [ZIPCODE="code"]
 [POSTCODE="code"]
 [COUNTRY="name"]
 [PHONE="num"]
 [ALTPHONE="num"]
 [FAX="num"]
 [EMAIL="addr"]
 [CONTACTUSERREF="username"]
 [LANGUAGE="name"]
 [LOGOFILE="file"]
 [LOGOTYPE="GIF|JPEG|TEXT"]>
</SPONSOR>
```

### Attributes

**NAME="name"** Name of the Sponsor. Optional.

**PROGRAM="text"**

Name of the trial. Optional.

**THERAPEUTICAREA="text"**

Therapeutic area of the trial. Optional.

**NOTE="text"**

Description of the trial. Optional.

**ADDRESS="addr1"**

First line of the Sponsor's address. Optional.

**ADDRESS2="addr2"**

Second line of the Sponsor's address. Optional.

**CITY="name"**

City in which the Sponsor's address is located. Optional.

**STATE="name"**

State in which the Sponsor's address is located. Optional.

**PROVINCE="name"**

Province in which the Sponsor's address is located. Optional.

**ZIPCODE="code"**

Sponsor's Zip code. Optional.

**POSTCODE="code"**

Sponsor's postal code. Optional.

**COUNTRY="name"**

Country in which the Sponsor's address is located. Optional.

**PHONE="num"**

Sponsor's telephone number. Optional.

**ALTPHONE="num"**

Sponsor's alternate telephone number. Optional.

**FAX="num"**

Sponsor's fax number. Optional.

**EMAIL="addr"**

E-mail address used for contacting the Sponsor. Optional.

**CONTACTUSERREF="username"**

Name of the user who is the primary Sponsor contact. Optional.

**LANGUAGE="name"**

Language used in correspondence with the Sponsor. Optional. English is the default.

**LOGOFILE="file"**

Filename of the Sponsor's logo file. Optional.

**LOGOTYPE="GIF|JPEG|TEXT"**

File type of the Sponsor's logo file: GIF, JPEG, or TEXT. Required if you specify a Logofile name.

## Example

This example illustrates the definition for a Sponsor called Acme Pharma.

```
<SPONSOR NAME="Acme Pharma"
 PROGRAM="AP603 Study"
 NOTE="New Drug for Hypertension"
 ADDRESS="123 Acme Court"
 ADDRESS2="Building 3-A"
 CITY="Western"
 STATE="Massachusetts"
 ZIPCODE="01240"
 PHONE="413-555-6666"
 FAX="413-555-7777"
 EMAIL="jsmith@acmepharma.com"
 CONTACTUSERREF="jsmith"
 LOGOFILE="acmelogo.gif"
 LOGOTYPE="GIF"/>
```

## StudyVersionDoc

### Purpose

The StudyVersionDoc tag enables you to assign a document, as defined in a *Documentation* (on page 113) tag, to a *StudyVersion* (on page 206).

### Syntax

```
<STUDYVERSIONDOC
 VERSIONDESCRIPTION="n"
 DOCREFNAME="name"
 [ORDER="n"]/>
```

### Attributes

**VERSIONDESCRIPTION="*n*"** Number of the *StudyVersion* (on page 206) with which to associate the document. Required.

**DOCREFNAME="*name*"**

RefName of the document, as specified in the *Documentation* (on page 113) tag. Required.

**[ORDER="*n*"]**

Order in which the tab representing the document will appear in the Document or Help window. Optional; the default is the order in which the document is referenced in the .xml file that defines StudyVersionDocs.

## Example

The following excerpt from a MEDMLDATA definition file illustrates the use of the StudyVersionDoc tag to set the *StudyVersion* (on page 206) of a set of documents to 1.

```
<MEDMLDATA>

<!-- Documents -->
<STUDYVERSIONDOC VERSIONDESCRIPTION="1" DOCREFNAME="Protocol"
ORDER="1"/>
<STUDYVERSIONDOC VERSIONDESCRIPTION="1" DOCREFNAME="Study"
ORDER="2"/>
<STUDYVERSIONDOC VERSIONDESCRIPTION="1" DOCREFNAME="Visit"
ORDER="3"/>
<STUDYVERSIONDOC VERSIONDESCRIPTION="1" DOCREFNAME="CRB"
ORDER="4"/>
<!-- System Help -->
<STUDYVERSIONDOC VERSIONDESCRIPTION="1" DOCREFNAME="AboutInForm"
ORDER="1"/>
<STUDYVERSIONDOC VERSIONDESCRIPTION="1" DOCREFNAME="GettingHelp"
ORDER="2"/>
<STUDYVERSIONDOC VERSIONDESCRIPTION="1" DOCREFNAME="UsingInForm"
ORDER="3"/>
<STUDYVERSIONDOC VERSIONDESCRIPTION="1" DOCREFNAME="ReportDesc"
ORDER="4"/>
<STUDYVERSIONDOC VERSIONDESCRIPTION="1" DOCREFNAME="ScreenDesc"
ORDER="5"/>
<STUDYVERSIONDOC VERSIONDESCRIPTION="1" DOCREFNAME="AdminHelp"
ORDER="6"/>

</MEDMLDATA>
```

## StudyVersionSite

### Purpose

The StudyVersionSite component records which StudyVersion a site is currently using. This component needs to be updated each time a StudyVersion changes in a way that does not require IRB approval and whenever IRB approval is received for a StudyVersion change that does require approval.

### Syntax

```
<STUDYVERSIONSITE
VERSIONDESCRIPTION="n"
[SITENAME="name"]
[SITEMNEMONIC="name"]
[ACCEPTDATE="date"]/>
```

## Attributes

**VERSIONDESCRIPTION="n"** Version number of the *StudyVersion* (on page 206) currently in use at the site. Required. This number must match the Version attribute with which the StudyVersion is defined.

**SITENAME="name"**

Name of the site. Either the Sitename or the Sitemnemonic attribute is required. This name must match the Name attribute with which the *Site* (on page 198) is defined.

**SITEMNEMONIC="name"**

Abbreviated name of the site. This name must match the Mnemonic attribute with which the *Site* (on page 198) is defined. Either the Sitename or the Sitemnemonic attribute is required.

**ACCEPTDATE="date"**

Date of IRB approval of a StudyVersion change.

**Note:** The Acceptdate attribute is for documentation purposes. The actual date that a site moves to a new StudyVersion is the date that the StudyVersionSite component for the site is revised in the database.

## Example

This example illustrates a set of StudyVersionSite components, one for each of five sites.

```
<STUDYVERSIONSITE VERSIONDESCRIPTION="2" SITEMNEMONIC="PF"
ACCEPTDATE="6/30/1998" />
<STUDYVERSIONSITE VERSIONDESCRIPTION="1" SITEMNEMONIC="BID"
ACCEPTDATE="6/30/1998" />
<STUDYVERSIONSITE VERSIONDESCRIPTION="2" SITEMNEMONIC="BCH"
ACCEPTDATE="6/30/1998" />
<STUDYVERSIONSITE VERSIONDESCRIPTION="2" SITEMNEMONIC="MGH"
ACCEPTDATE="6/30/1998" />
<STUDYVERSIONSITE VERSIONDESCRIPTION="2" SITEMNEMONIC="BWH"
ACCEPTDATE="6/30/1998" />
```

## StudyVersion

### Purpose

The StudyVersion tag groups *FormSets* (see "*FormSet*" on page 127) together under a single version of a trial and its protocol. Each time you need to implement a revised form or trial document, you create a new StudyVersion definition, update the *StudyVersionSite* (on page 205) definition for each site where the changed form version applies, and update the *StudyVersionDoc* (on page 204) definition for each applicable trial document.

## Syntax

```
<STUDYVERSION
 UUID="id"
 STUDYNAME="name"
 VERSIONDESCRIPTION="text"
 [PROTOCOL="name"]
 [SPONSORDATE="date"]
 [TRADEDRUGNAME="name"]
 [GENERICDRUGNAME="name"]
 [SPONSORDRUGNAME="name"]>
 <FORMSET+ attributes/>
</STUDYVERSION>
```

### Attributes

**UUID="id"** *Universally Unique Identifier* (see "*MedML Schema*" on page 11); a string that identifies the component uniquely across all trials, trial databases, and machines. Optional.

**STUDYNAME="name"**

Name of the trial. Required.

**VERSIONDESCRIPTION="text"**

Description of the StudyVersion; for example, an indication of what was changed in this version. Required.

**PROTOCOL="name"**

Name and version number of the IRB-approved trial protocol. Optional.

**SPONSORDATE="date"**

Date of sponsor approval of the StudyVersion, if applicable. Optional.

**TRADEDRUGNAME="name"**

Trade name of the drug being studied. Optional.

**GENERICDRUGNAME="name"**

Generic name of the drug being studied. Optional.

**SPONSORDRUGNAME="name"**

Sponsor's name for the drug being studied. Optional.

## Children

A StudyVersion definition must include one or more *FormSet* (on page 127) definitions. Each *FormSet* (on page 127) defines a visit or other group of *Forms* (see "*Form*" on page 122) and specifies which Forms to include in the set. Because FormSets are not reusable, they are defined inside the definition of a StudyVersion.

**Warning:** If the StudyVersion definition includes one or more *common CRFs* (see "*FormSet*" on page 127), consider that once you define a common CRF by including it in a COMMONCRF FormSet, and data for any patient has been entered in it, you cannot revert the form to regular CRF status by removing it from the COMMONCRF FormSet in the *StudyVersion* (on page 206) definition. Similarly, you cannot change a CRF in a regular VISIT FormSet into a common CRF by adding it to a COMMONCRF FormSet after it contains data for any patient. If you attempt to change a StudyVersion in either of these ways, you will lose patient data.

If you need to create a regular CRF that captures the same data as an existing common CRF, create it as a separate *Form* (on page 122) definition with a different REFNAME from the common CRF, and add it to the appropriate VISIT FormSets in the StudyVersion.

## Example

The following example illustrates the second version of a StudyVersion definition that contains four FormSets representing:

- Visits on Week -4, Week -2, and Week 0
- A screening set containing a Screening Log

```
<STUDYVERSION
 VERSIONDESCRIPTION="2"
 STUDYNAME="Hypertension Study"
 PROTOCOL="Protocol XYZZY">
 <FORMSET REFNAME="Visit1" TITLE="Week -4"
 LANGUAGE="English"TYPE="Visit"
 SCHEDULED="true"ORDER="1">
 <FORMREF REFNAME="DEM"/>
 <FORMREF REFNAME="FH"/>
 </FORMSET>
 <FORMSET REFNAME="Visit2" TITLE="Week -2"
 TYPE="Visit"SCHEDULED="true">
 <FORMREF REFNAME="VS"/>
 </FORMSET>
 <FORMSET REFNAME="Visit3" TITLE="Week 0"
 TYPE="Visit"SCHEDULED="true">
 <FORMREF REFNAME="VS"/>
 </FORMSET>
 <FORMSET REFNAME="screen" TITLE="Screening Log"
 UUID="d882ce38-0f42-11d2-a419-00a0c963e0ac"
 TYPE="SCREENING"SCHEDULED="false">
 <FORMREF REFNAME="screen"/>
 </FORMSET>
```



## SysConfig

### Purpose

A SysConfig component specifies the setting of an InForm system configuration variable.

### Syntax

```
<SYSCONFIG
 CONFIGNAME="name"
 TYPE="0"
 VALUE="n"/>
```

### Attributes

**CONFIGNAME="name"** Name of the configuration variable. Required. The InForm application has the following configuration variables:

- AllowCPResize—1 (yes) or 0 (no), indicating whether the InForm system permits a user to minimize the navigation panel on the left side of the screen, including the user's picture. The default is 0.
- AllowPasswordReuse—1 (yes) or 0 (no), indicating whether users can change to a previously used password when performing password updates. The default is 1.
- AllowedRuleObjects—Specifies the scripting objects that can be called by a user-defined rule or execution plan. The default objects are CDO.Message and CDONTS.NewMail.

**Note:** You must change the HKEY\_LOCAL\_MACHINE\SOFTWARE\Phase Forward\InForm\WhiteListScriptObjects registry key to 0 (zero) before you can update the allowed rules object list. Before modifying the list of allowed rule objects, contact Oracle Global Support for assistance.

- AutoAnswerManualQueries—1 (on) or 0 (off), indicating whether the InForm system automatically answers a manual query when a data item change satisfies the rules on the data item. The default is 1.
- CookServer—Name of the server used for installing MedML metadata definitions in a configuration that uses the InForm Unplugged.
- DaysPasswordExpiration—Number of days that can pass before the InForm system requires users to change their passwords. The default is 30.
- DefaultCPMaximized — 1 (yes) or 0 (no), indicating whether the Navigation pane in the InForm application window opens maximized or not. The default is 1.
- EnableForgotPassword—1 (yes) or 0 (no), indicating whether to enable the feature that lets users request a password reset if they have forgotten their password. The default is 1.
- EmailForForgotPasswordNotification—The email address of an administrator who receives notification when a user requests a password reset.
- EmailForNewSiteAndUserNotification—The email address of an administrator who receives notification when a new site or new user is added.

- EnforceVisitDate—1 (yes) or 0 (no), indicating whether to require the use of Date of Visit on the first form of every visit. The default is 0.
- EnrollWithIncompleteForms—1 (yes) or 0 (no), indicating whether the InForm system permits a patient to be enrolled with incomplete screening or enrollment information, after override authorization. The default is 0.
- ExePlanServer—The name of the server(s) defined as the server(s) on which execution plans run in a configuration that uses the InForm Unplugged.
- FORMSAVEMODE—The location and format of the "Form submitted successfully" message.
  - Inline: message displays in the header of the form.
  - Popup: message displays as a pop-up. User must click OK in order to proceed. Default.
- InactivateRetryCount—Number of failed login attempts to allow before inactivating the user account. The default is 3.
- INLINEDURATION—1-9, Specifies the number of seconds that the "Form submitted successfully" message remains visible in the header of the form before it fades. Applies when FORMSAVEMODE is set to Inline.
- JUMPTOCREATEQUERY—1 (yes) or 0 (no), if set to Yes, clicking an item's grey query flag displays the Create Query form.
- MaxNumOfResubmissions—Maximum number of times to retry submission of a failed execution plan before it is logged as an error in the event log and removed from the queue of execution plans to be run. The default is 2.
- MinPasswordLength—Minimum number of characters required for passwords. The default is 6.
- MinutesReauthenticate—Number of minutes of inactivity that can pass before the InForm system requires a user to log in again. The default is 5.
- MinutesReIdentification—Number of minutes that a session can be active before the InForm system requires a user to log in again. The default is 120.
- NavigationMode—1 (enable) or 0 (disable), indicating whether to enable or disable special navigation modes for a trial. The default is 0.
- NumCharsCRFLength—Maximum number of lines on a CRF to create a readable PDF. The default is 100.
- NumOfExePlanListenThreads—Number of threads running in the background to process pending execution plans. The default is 4; at least 1 is required for any execution plans to run.
- OneNonAlphaNumericCharacter—1 (yes) or 0 (no), indicating whether passwords must include at least one special character. The default is 0.
- OneNumericalCharacter—1 (yes) or 0 (no), indicating whether passwords must include at least one numeric character. The default is 0.
- OneUppercaseCharacter—1 (yes) or 0 (no), indicating whether passwords must include at least one uppercase character. The default is 0.
- PatientSequence—Format for assigning patient numbers. For information about sequence number formats, the *Setting Up a Trial with InForm Architect and MedML Guide*.

- PostQueryForConflictResolution—Not supported.
- QueryMaxLength—Maximum number of characters of query text displayed below an item on a CRF. The default is 80.
- QUERYSELECTION—Specifies whether queries are created in Opened or Candidate state.
- RandCentralStratified—Sequence number format for Central Stratified randomization schemes (multiple drug kit lists, patient assigned a drug kit based on stratification criteria). For information about sequence number formats, see the *Setting Up a Trial with InForm Architect and MedML Guide*.
- RandomizationSrc—Name of the randomization source manager (COM object) that accesses the default randomization source database. The default name is Inform.PFRandomization.1.
- RandSimpleCentral—Sequence number format for Simple Central randomization schemes (one central drug kit list from which numbers are assigned sequentially). For information about sequence number formats, see the *Setting Up a Trial with InForm Architect and MedML Guide*.
- RandSimpleCentralSRC—Name of the randomization source manager (COM object) that accesses the randomization source database for Simple Central randomization schemes.
- RandSimpleSite—Sequence number format for Simple Site randomization schemes (drug kit list for each site, patient assigned sequentially to drug kit on list for their site). For information about sequence number formats, see *Setting Up a Trial with InForm Architect and MedML Guide*.
- RandSimpleSiteSRC—Name of the randomization source manager (COM object) that accesses the Simple Site randomization source database.
- RandStratifiedBySite—Sequence number format for Site Stratification randomization schemes (multiple drug kit lists for each site, patient assigned a drug kit based on site and stratification criteria). For information about sequence number formats, see *Setting up a Trial with InForm Architect Software and MedML*.
- RandStratifiedBySiteSRC—Name of the randomization source manager (COM object) that accesses the Site Stratification randomization source database.
- RequireCommentForNA—1 (yes) or 0 (no), indicating whether the InForm system requires a user to enter a comment when entering N/A, Unknown, or Not Done in response to a question on a form. The default is 0.
- ScreeningSequence—Sequence number format for assigning screening numbers. For information about sequence number formats, see *Setting up a Trial with InForm Architect Software and MedML*.
- SponsorEditFrozen—1 (yes) or 0 (no), indicating whether sponsors will be able to edit a form after it has been marked as frozen. The default is 0.
- SSLFlag—1 (on) or 0 (off), indicating whether HTTPS should be enabled to provide encryption of data. The default is 0.

**Note:** Before this option can take effect, you must stop and restart the trial.

- SDVSTICKYMODE—1 (enabled) or 2 (disabled), if Enabled, a button to enable or disable Source Verification mode appears on all forms, for users who have Source Verification rights. Default is Disabled.
- TrialDateFormat—"Month\_Day\_Year", "Day\_Month\_Year" or "Year\_Month\_Day", indicating the format in which you want the date to appear in the trial. The default is Month\_Day\_Year.
- UNC\_DownloadDirectory—Full path name of the physical InForm application server directory used to download data listings with the Listings button.
- UniqueIntlDOBSwch—Trial, site or none, indicating whether the InForm system requires a unique combination of patient initials and date of birth for a trial, a site or not at all:
  - 0 (default)—Initials and DOB combination is not required to be unique.
  - 1—Initials and DOB combination must be unique within a site.
  - 2—Initials and DOB combination must be unique within a trial.

**Note:** If you specify that unique initials and date of birth is not required and patients with duplicate initials and date of birth are entered and then you specify that unique IDs are required, the previously entered duplicate information will not be reported. **Patient record transfer consideration:** If you plan to allow the transfer of patients from one site to another, be aware that if a user transfers a patient to a site where another patient exists with the same initials and date of birth, and the trial does not require unique initials and date of birth or only requires site uniqueness, the patient transfer fails. The user must change the patient initials to make the combination unique.

To prevent this situation, set the UniqueIntlDOBSwch attribute to require unique initials and DOB across the trial.

- UniquePatIDSwch—Trial, site or none, indicating whether the InForm system requires a unique patient ID for a trial, a site or not at all:
  - 0 (default)—Patient ID is not required to be unique.
  - 1—Patient ID must be unique within a site.
  - 2—Patient ID must be unique within a trial.

**Note:** If you specify that a unique patient number is not required and patients with identical numbers are entered, and then you specify that unique patient numbers are required, the previously entered duplicate information will not be reported. **Patient record transfer consideration:** If you plan to allow the transfer of patients from one site to another, it is highly recommended that you require patient numbers to be unique across the entire trial. InForm application does not allow a patient to be transferred to another site in which a patient exists with the same patient number. If a conflict arises, InForm application rejects the transfer, and you must manually change the patient number of the transferring patient to a value that is not duplicated at the target site.

- `USERTYPE="SYSTEM|SITE|SPONSOR|SUPPORT"`

Type of user.

- **SYSTEM**—User with specialized system capabilities. For example, when the InForm application generates a query automatically, the user name assigned as the query originator is "autoquery." The autoquery user is a system user.
- **SITE**—User associated with a site.
- **SPONSOR**—User associated with a sponsor.
- **SUPPORT**—User typically responsible for support or troubleshooting tasks. Support users behave like all other InForm sponsor users, except they cannot edit their user name or user type.

Required.

- `ViewCRFSignList` — 1 (yes) or 0 (no), indicating whether a list of required signatures should appear on each CRF for which a signature is required. The default is 1.
- `Virtual_DownloadDirectory`—Full path name of the InForm application server virtual directory used to download data listings with the Listings button.

**Note:** You can set the **Server Friendly Name** parameter, which provides a user-friendly server name to be displayed on **Query Details** and **Signing Details** screens, only through the **Admin user interface** of InForm application. There is no equivalent MedML variable.

**TYPE="0"**

0 is the only value currently accepted. Required.

**VALUE="n"**

Value to assign to the configuration variable. Required.

## Example

The following example shows the XML tags used to insert the default configuration parameter values into the database.

```
<SYSCONFIG CONFIGNAME="MinutesReauthenticate" TYPE="0" VALUE="5"/>
<SYSCONFIG CONFIGNAME="DaysPasswordExpiration" TYPE="0" VALUE="30"/>
<SYSCONFIG CONFIGNAME="MinutesReIdentification" TYPE="0" VALUE="120"/>
<SYSCONFIG CONFIGNAME="MinPasswordLength" TYPE="0" VALUE="6"/>
<SYSCONFIG CONFIGNAME="InactivateRetryCount" TYPE="0" VALUE="3"/>
<SYSCONFIG CONFIGNAME="OneNumericalCharacter" TYPE="0" VALUE="0"/>
<SYSCONFIG CONFIGNAME="OneUppercaseCharacter" TYPE="0" VALUE="0"/>
<SYSCONFIG CONFIGNAME="OneNonAlphaNumericCharacter" TYPE="0" VALUE="0"/>
<SYSCONFIG CONFIGNAME="AllowPasswordReuse" TYPE="0" VALUE="1"/>
<SYSCONFIG CONFIGNAME="EnableForgotPassword" TYPE="0" VALUE="1"/>
<SYSCONFIG CONFIGNAME="EmailForForgotPasswordNotification" TYPE="0"
VALUE=""/>
<SYSCONFIG CONFIGNAME="EmailForNewSiteAndUserNotification" TYPE="0"
VALUE=""/>
<SYSCONFIG CONFIGNAME="EnrollWithIncompleteForms" TYPE="0" VALUE="0"/>
<SYSCONFIG CONFIGNAME="NumCharsCRFLength" TYPE="0" VALUE="100"/>
<SYSCONFIG CONFIGNAME="NumOfExePlanListenThreads" TYPE="0" VALUE="4"/>
```

```

<SYSCONFIG CONFIGNAME="MaxNumOfResubmissions" TYPE="0" VALUE="2"/>
<SYSCONFIG CONFIGNAME="RequireCommentForNA" TYPE="0" VALUE="0"/>
<SYSCONFIG CONFIGNAME="AllowCPResize" TYPE="0" VALUE="0"/>
<SYSCONFIG CONFIGNAME="SSLFlag" TYPE="0" VALUE="1"/>
<SYSCONFIG CONFIGNAME="RandomizationSrc" TYPE="0"
VALUE="Inform.PFRandomization.1" />
<SYSCONFIG CONFIGNAME="RandSimpleCentral" TYPE="0" VALUE="SC:RND-%q"/>
<SYSCONFIG CONFIGNAME="AutoAnswerManualQueries" TYPE="0" VALUE="1"/>
<SYSCONFIG CONFIGNAME="QueryMaxLength" TYPE="0" VALUE="80"/>
<SYSCONFIG CONFIGNAME="EnforceVisitDate" TYPE="0" VALUE="0"/>
<SYSCONFIG CONFIGNAME="DEFAULTCPMAXIMIZED" VALUE="1"/>
<SYSCONFIG CONFIGNAME="SPONSOREDITFROZEN" VALUE="0"/>
<SYSCONFIG CONFIGNAME="UNIQUEPATIDSWTCH" VALUE="0"/>
<SYSCONFIG CONFIGNAME="UNIQUEINTLDOBSWTCH" VALUE="0"/>
<SYSCONFIG CONFIGNAME="PostQueryForConflictResolution" VALUE="1"/>
<SYSCONFIG CONFIGNAME="TrialDateFormat" VALUE="MONTH_DAY_YEAR"/>
<SYSCONFIG CONFIGNAME="NavigationMode" VALUE="0"/>
<SYSCONFIG CONFIGNAME="FORMSAVEMODE" VALUE="1"/>
<SYSCONFIG CONFIGNAME="INLINEDURATION" VALUE="3"/>
<SYSCONFIG CONFIGNAME="ViewCRFSignList" VALUE="1"/>
<SYSCONFIG CONFIGNAME="SDVSTICKYMODE" TYPE="0" VALUE="0"/>
<SYSCONFIG CONFIGNAME="JUMPTOCREATEQUERY" VALUE="0"/>
<SYSCONFIG CONFIGNAME="QUERYSELECTION"
VALUE="QUERYACTION_CREATEOPEN"/>

```

## Testcase

### Purpose

The Testcase tag enables you to specify the values to be substituted for the arguments in a rule definition when the rule is tested in the InForm Architect application, along with the expected test result. The Testcase tag is included as a child component of the **Rule** (on page 178) tag.

### Syntax

```

<TESTCASE
 [RESULT="Pass | Fail"]
 [INPUT_1="value"]
 [INPUT_2="value"]
 [INPUT_3="value"]
 [INPUT_4="value"]
 [INPUT_5="value"]
 [INPUT_6="value"]
 [INPUT_7="value"]
 [INPUT_8="value"]
 [INPUT_9="value"]/>

```

## Attributes

**RESULT="Pass|Fail"** Indicates whether the rule should pass or fail when it is run using the values supplied as arguments in the INPUT\_1 through INPUT\_9 attributes. Values are Pass or Fail; Optional.

```
INPUT_1="value"
INPUT_2="value"
INPUT_3="value"
INPUT_4="value"
INPUT_5="value"
INPUT_6="value"
INPUT_7="value"
INPUT_8="value"
INPUT_9="value"
```

Each of these attributes specifies a value to be passed into the rule as an argument when testing the rule in the InForm Architect application. Enter the values in the order in which the arguments occur in the rule; use as many INPUT attributes as there are arguments.

## Example

In the following Rule definition, *Rulearg* (on page 182) tags are used to define the arguments min, max, qtext, and addpath. Four values are supplied as test cases with the *Testcase* (on page 214) tag.

```
<RULE REFNAME="rulRangeCheck"
 DESCRIPTION="Ensure data are within upper and lower range"
 ENABLED="true"
 SCRIPTTYPE="SERVERRULE">
<![CDATA[Name : rulRangeCheck
'Desc :Compares minimum and maximum ranges entered to ensure that
'data are within the expected range. This is a generic rule that can
'be attached to any item that has an upper and lower range.
'Args:
'min - minimum value
'max - maximum value
'qtext - querytext
'addpath - additional path to the control (beyond the item)
'
```

Option Explicit

Dim Min, Max, qtext, addpath, item, v\_val

```
Min = Patient.GetArgument("min")
Max = Patient.GetArgument("max")
qtext = Patient.GetArgument("qtext")
addpath = Patient.GetArgument("addpath")
item = Patient.GetCurPath() & addpath
```

```
Patient.AddNamedValue "QUERYTEXT", qtext
```

```
Result.Rulepassed = 1>true
```

```
v_val = Patient.GetValue(item, "", 0,0,0)
```

```
 If cDbl(v_val) < cDbl(Min) or cDbl(v_val) > cDbl(Max) then
```

```
 Result.Rulepassed = 0
 End If
]]>
<EVENTREF REFNAME="evtGeneric"/>
<RULEARG
 NAME="addpath"
 TYPE="STRING"/>
<RULEARG
 NAME="qtext"
 TYPE="STRING"/>
<RULEARG
 NAME="min"
 TYPE="STRING"/>
<RULEARG
 NAME="max"
 TYPE="STRING"/>
<TESTCASE
 RESULT="Pass"
 INPUT_1="15"
 INPUT_2="22"
 INPUT_3="93"
 INPUT_4="2"/>
</RULE>
```

## TextControl

### Purpose

The TextControl tag defines a text box into which a user can type data. By using *Controlrefs* (see "*Controlref*" on page 56), you can include TextControl definitions in the definitions of the following types of form components:

- *CheckBoxControl* (on page 95)
- *GroupControl* (on page 134)
- *Item* (on page 139)
- *RadioControl* (on page 163)

**Note:** When defining compound controls with *Controlref* (on page 56) definitions, ensure that all subordinate controls return the same data type, by defining them with the same TYPE attribute.

Additionally, when defining compound controls, note that the InForm application supports a maximum of five levels of nesting. Although five levels are supported, as a design practice, you should attempt to minimize the number of nested levels to help performance.



## Syntax

```
<TEXTCONTROL
 REFNAME="name"
 [DESIGNNOTE="text"]
 [NAME="name"]
 [UUID="id"]
 [LANGUAGE="name"]
 [CAPTION="text"]
 [CAPTIONALIGN="LEFT|RIGHT|TOP|BOTTOM"]
 [ALIGN="LEFT|CENTER|RIGHT|TOP|MIDDLE|BOTTOM"]
 [HEIGHT="n"]
 LENGTH="n"
 [MAXLENGTH="n"]
 [DATATYPE="STRING|INTEGER|FLOAT|PASSWORD"]>
 [PRECISION="n"]
 [MINVALUE="n"]
 [MINPROPERTY="GREATERTHAN|GREATERTHANEQUAL"]
 [MAXVALUE="n"]
 [MAXPROPERTY="LESSTHAN|LESSTHANEQUAL"]
 [UNITDISPLAYTYPE="PULLDOWN|RADIO|ELEMENT">]
 <UNTREF* attributes/>
</TEXTCONTROL>
```

### Attributes

**REFNAME="name"** Name used when referring to the TextControl in the definition of another control. Required. This name must be unique among TextControls.

### DESIGNNOTE="text"

Free-form text, with a maximum of 255 characters, containing any information you want to capture about the design of the component. This information is for documentation only and is not displayed. Optional.

### NAME="name"

Name used when referring to the TextControl in the definition of another control. Optional.

### UUID="id"

*Universally Unique Identifier* (see "*MedML Schema*" on page 11); a string that identifies the component uniquely across all trials, trial databases, and machines. Optional.

### LANGUAGE="name"

Language used to display the caption. Optional; English is the default.

### CAPTION="text"

Text that appears on the screen with the text box. Optional.

### CAPTIONALIGN="LEFT|RIGHT|TOP|BOTTOM"

Position of the caption relative to the text control: Left, Right, Top, or Bottom. Optional; Left is the default.

**ALIGN="LEFT | CENTER | RIGHT | TOP | MIDDLE | BOTTOM"**

Alignment of the UnitRef components within the control: Left, Center, Right, Top, Middle, or Bottom. Optional; Left is the default.

**HEIGHT="n"**

Height of the text box in number of lines. Optional. 1 is the default.

**LENGTH="n"**

Length of the text box in number of characters. Required.

**MAXLENGTH="n"**

Maximum length allowed for entered text. Optional. The value specified for Length is the default.

**DATATYPE="STRING | INTEGER | FLOAT | PASSWORD"**

Type of data expected in the text box: String, Integer, Float, or Password. Optional; String is the default.

**Note:** If you use Integer, the length must be less than 10 characters.  
Password is used only internally on InForm system forms.

**PRECISION="n"**

For values with a DATATYPE of FLOAT, the number of digits expected to be entered after the decimal point. This value must be between 0 and 15, where zero equals no digits after the decimal point. Optional.

**MINVALUE="n"**

For numeric DATATYPES, a value used to test the minimum allowable range of a data item; used in conjunction with the MINPROPERTY attribute. Optional.

**MINPROPERTY="GREATERTHAN | GREATERTHANEQUAL"**

For numeric DATATYPES, an operator used to test the value entered in the control; used in conjunction with the MINVALUE attribute:

- GREATERTHAN — The entered value must be greater than the specified MINVALUE.
- GREATERTHANEQUAL — The entered value must be greater than or equal to the specified MINVALUE.

Optional. For example, if you enter 50 as the MINVALUE and GREATERTHAN as the MINPROPERTY, a data item must have a value of 50 or more, or the InForm application generates an error message. The use of the MINVALUE and MINPROPERTY attributes takes the place of a range- checking rule on the item. **MAXVALUE="n"**

For numeric DATATYPES, a value used to test the maximum allowable range of a data item; used in conjunction with the MAXPROPERTY attribute. Optional.

**MAXPROPERTY="LESSTHAN|LESSTHANEQUAL"**

For numeric DATATYPES, an operator used to test the value entered in the control; used in conjunction with the MAXVALUE attribute:

- LESSTHAN — The entered value must be less than the specified MAXVALUE.
- LESSTHANEQUAL — The entered value must be less than or equal to the specified MAXVALUE.

Optional. For example, if you enter 100 as the MAXVALUE and LESSTHAN as the MAXPROPERTY, a data item must have a value of less than 100, or the InForm application generates an error message. The use of the MAXVALUE and MAXPROPERTY attributes takes the place of a range-checking rule on the item.

**UNITDISPLAYTYPE="PULLDOWN|RADIO|ELEMENT"**

Type of control used to display *Units* (see "*Unit*" on page 220) included in the TextControl with a *Unitref* (on page 223) definition: pulldown list, radio button list, or simple component. Element is the default.

**Children**

A TextControl definition can include zero or more *Unitref* (on page 223) definitions. Each Unitref refers to a previously defined *Unit* (on page 220).

**Example**

This TextControl defines a text box in which a user can enter a two-line comment:

If clinically significant, please comment:

```
<TEXTCONTROL REFNAME="COMMENTBOX"
 NAME="COMMENTBOX"
 CAPTION="If clinically significant, please comment: "
 CAPTIONALIGN="TOP"
 HEIGHT="2"
 LENGTH="40"
 MAXLENGTH="40"
 DATATYPE="STRING"/>
```

## Unit

### Purpose

The Unit tag specifies a type of unit in which the user can enter data. The Unit definition includes a reference to a conversion rule that enables the InForm application to convert to a standard unit for reporting or statistical analysis.

### Syntax

```
<UNIT
 REFNAME="name"
 [DESIGNNOTE="text"]
 [UUID="id"]
 SYMBOL="text"
 CLASSIFICATION="text"
 [LANGUAGE="name"]
 BASEREFNAME="name"
 [CONVERSIONTOBASE="name"]
 [CONVERSIONFROMBASE="name"]/>
```

### Attributes

#### **REFNAME="name"**

Name used when referring to the Unit in the definition of another control. Required. This name must be unique among Units.

**Note:** A *TextControl* (on page 216) defined with TYPE="STRING" cannot have a Unit. A TextControl defined with TYPE="INTEGER" or "FLOAT" can have a Unit.

#### **DESIGNNOTE="text"**

Free-form text, with a maximum of 255 characters, containing any information you want to capture about the design of the component. This information is for documentation only and is not displayed. Optional.

**UUID="id"**

*Universally Unique Identifier* (see "*MedML Schema*" on page 11); a string that identifies the component uniquely across all trials, trial databases, and machines. Optional.

**Note:** The unit definitions that are included with the Base trial have predefined UUIDs that are not required but are listed in the following table for reference.

Unit	UUID
BPDIAS	PF_BP_DIAS
BPSYS	PF_BP_SYS
Celsius	498100ff-e9df-11d1-9e60- 00a0c9769a33
Centimeter	498100f5-e9df-11d1-9e60- 00a0c9769a33
Fahrenheit	498100fe-e9df-11d1-9e60- 00a0c9769a33
Inches	498100f2-e9df-11d1-9e60- 00a0c9769a33
Kilogram	498100f8-e9df-11d1-9e60- 00a0c9769a33
Pound	498100fc-e9df-11d1-9e60- 00a0c9769a33

**SYMBOL="text"**

Label used to represent the unit on the screen. Required.

**CLASSIFICATION="text"**

Type of measurement represented by the unit. Required. The following classifications have been predefined:

- Length
- Weight
- Temperature
- Pressure
- Volume
- Area
- Time
- Frequency

**LANGUAGE="name"** Language used to display the label. Optional; English is the default.

**BASEREFNAME="name"**

RefName of the component used as a basis for conversion. Required.

**CONVERSIONTOBASE="name"**

RefName of a VBScript conversion rule used to convert the entered value to the base value.  
Optional.

**CONVERSIONFROMBASE="name"**

RefName of a VBScript conversion rule used to convert to this Unit from the Unit specified in the Baserefname attribute. Not currently used.

## Examples

### Example 1

This example defines a unit called "Inches." In this example, inches are both the Unit being defined and the Unit specified as the basis for conversion, so the conversion rule specified in the CONVERSIONTOBASE attribute simply multiplies the entered value by 1:

```
<UNIT REFNAME="INCHES" SYMBOL="IN" CLASSIFICATION="Length"
 BASEREFNAME="Inches" CONVERSIONTOBASE="UnitIsBase"
 UUID="498100f2-e9df-11d1-9e60-00a0c9769a33"/>
```

UnitIsBase conversion rule:

```
Data.Result=Data.BaseValue*1
```

### Example 2

This example defines a unit called "Centimeters." In this example, the Unit specified as the basis for conversion is "Inches." Therefore, the conversion rule referenced in the Conversiontobase attribute reflects the factor used to convert to inches from centimeters.

```
<UNIT REFNAME="CENTIMETERS" SYMBOL="CM" CLASSIFICATION="Length"
 BASEREFNAME="Inches" CONVERSIONTOBASE="CmToInches"
 UUID="498100f5-e9df-11d1-9e60-00a0c9769a33"/>
```

CmToInches conversion rule:

```
Data.Result=Data.Basevalue*.3937
```

## Unitref

### Purpose

The Unitref tag enables you to include the definition of a *Unit* (on page 220) in the definition of another control. A Unitref appears only as the child of the control in which it is included; it is not submitted as a stand-alone component. You can include a Unitref in any of the following types of controls:

- *CalculatedControl* (on page 93)
- *CheckBoxControl* (on page 95)
- *GroupControl* (on page 134)
- *PullDownControl* (on page 158)
- *RadioControl* (on page 163)
- *SimpleControl* (on page 196)
- *TextControl* (on page 216)

### Syntax

```
<UNITREF
 REFNAME="name"/>
```

### Attributes

**REFNAME="name"** RefName of the *Unit* (on page 220) that the Unitref is including in the control definition. Required.

**Note:** When defining compound controls with *Controlref* (on page 56) or Unitref definitions, ensure that all subordinate controls return the same data type, by defining them with the same **TYPE** attribute.

Additionally, when defining compound controls, note that the InForm application supports a maximum of five levels of nesting. Although five levels are supported, as a design practice, you should attempt to minimize the number of nested levels to help performance.

### Example

In this example, *Unit* (on page 220) definitions of pounds and kilograms are included in a *PullDownControl* (on page 158) definition by using Unitref tags.

- 1 Create *Unit* (on page 220) definitions for pounds and kilograms. In this example, kilograms are the base units for conversion.
- 2 

```
<UNIT REFNAME="KILOGRAMS" SYMBOL="KG" CLASSIFICATION="Weight"
 BASEREFNAME="KILOGRAMS" CONVERSIONTOBASE="UnitIsBase"
 UUID="498107f2-e9df-11d1-9e60- 00a0c9769a33"/>
<UNIT REFNAME="POUNDS" SYMBOL="LB" CLASSIFICATION="Weight"
 BASEREFNAME="KILOGRAMS" CONVERSIONTOBASE="LbToKg"
 UUID="498109f5-e9df-11d1-9e60- 00a0c9769a33"/>
```

- 3 Create a *PullDownControl* (on page 158) definition that uses Unitref tags to refer to the *Unit* (on page 220) definitions for pounds and kilograms. In this example, the unit selections are displayed in a pull-down list.
- 4 

```
<PULLDOWNCONTROL REFNAME="WEIGHTPULLDOWN"
NAME="WEIGHTPULL">
 <UNTREF REFNAME="KILOGRAMS"/>
 <UNTREF REFNAME="POUNDS"/>
</PULLDOWNCONTROL>
```

## Update\_Form\_Section

### Purpose

The Update\_Form\_Section tag is used to update a section within a form without modifying the StudyVersion. This XML is used to modify an existing section on a form. It cannot be used to add a new section. If patient data already exists for this section, it is linked to the old version of the section. Data entered after the update is linked to the newer version of the section.

When deleting items with this tag, keep in mind that the data will still exist in the database, but will not show up on the form. Instead of removing form items through this method, it's preferable to remove them by updating the Study Version for the trial.

### Syntax

```
<METADATA>
<UPDATE_FORM_SECTION
 FORM_REFNAME="name"
 FORM_REVISION="number"
 SECTION_REFNAME="name"
 SECTION_REVISION="number"/>
</METADATA>
```

### Attributes

**FORM\_REFNAME="name"** The RefName of the form you want to modify. Required.

**FORM\_REVISION="number"**

The revision number of the form you want to modify. Required.

**SECTION\_REFNAME="name"**

The RefName of the section you want to modify. Required.

**SECTION\_REVISION="number"**

The revision number of the section you want to modify. Required.



## Updating a section definition using the Update\_Form\_Section tag

- 1 Stop the trial and make sure that no one can access the trial or modify the trial data.
- 2 Make the necessary changes to the XML for the section you want to modify.
- 3 Install the new version of the section using the MedML Installer utility.
- 4 Generate the XML for the "UPDATE\_FORM\_SECTION." You can use SQL to retrieve the new and old section revision numbers or you can look in the database directly.
- 5 For FORM\_REVISION number, open the PF\_PAGE table and look up the latest (highest number) PAGEREVISIONNUMBER for the corresponding PAGEREFNAME. For SECTION\_REVISION number, open the PF\_SECTION table and look up the most recent (highest number) SECTIONREVISIONNUMBER for the corresponding SECTIONREFNAME.
- 6 Run the MedML Installer utility and apply the "UPDATE\_FORM\_SECTION" xml. The installer will validate that the form RefName and revision is correct, and that the form contains the section you updated.
- 7 Restart the trial.
- 8 Test the updated form version after applying this change. The MedML Installer utility does not validate the item metadata against any existing patient data.

## Update\_Section\_Item

### Purpose

The Update\_Section\_Item tag is used to update an item within a section without modifying the StudyVersion. This XML is used to modify an existing item. It cannot be used to add a new item. If patient data already exists for this item, it is linked to the old version of the item. Data entered after the update is linked to the newer version of the item.

**WARNING: Making changes to trial definitions without creating a new trial version violates Good Clinical Practice (GCP) and can cause problems with trial data.**

### Syntax

```
<METADATA>
<UPDATE_SECTION_ITEM
 SECTION_REFNAME="name"
 SECTION_REVISION="number"
 ITEM_REFNAME="name"
 ITEM_REVISION="number"/>
</METADATA>
```

**Attributes**

**SECTION\_REFNAME="name"** The RefName of the section you want to modify. Required.

**SECTION\_REVISION="number"**

The revision number of the section you want to modify. Required.

**ITEM\_REFNAME="name"**

The RefName of the item you want to modify. If the modified item is contained in an itemset, this RefName is the RefName of the itemset. Required.

**ITEM\_REVISION="number"**

The revision number of the item you want to modify. If the modified item is contained in an itemset, this revision number is the revision number of the itemsetRequired.

**Updating a section definition using the Update\_Section\_Item tag**

**WARNING: Making changes to trial definitions without creating a new trial version violates Good Clinical Practice (GCP) and can cause problems with trial data.**

- 1 Stop the trial and make sure that no one can access the trial or modify the trial data.
- 2 Make the necessary changes to the XML for the item you want to modify.
- 3 Install the new version of the item using the MedML Installer utility.
- 4 Generate the XML for the "UPDATE\_SECTION\_ITEM." You can use SQL to retrieve the revision numbers for the new and old items or you can look in the database directly.
- 5 For the SECTION\_REVISION number, open the PF\_SECTION table and look up the latest (highest number) SECTIONREVISIONNUMBER for the corresponding SECTIONREFNAME. For the ITEM\_REVISION number, open the PF\_ITEM table and look up the most recent (highest number) ITEMREVISIONNUMBER for the corresponding ITEMREFNAME.

**Note: If the item you are modifying is contained in an itemset, the RefName and revision number you enter here must be for the itemset. Itemset RefNames and revision numbers are also stored in the PF\_ITEM table.**

- 6 Run the MedML Installer utility and apply the "UPDATE\_SECTION\_ITEM" xml. The installer will validate that the form RefName and revision is correct, and that the form contains the item you updated.
- 7 Restart the trial.
- 8 Test the updated form version after applying this change. The MedML Installer utility does not validate the item metadata against any existing patient data.

## User

### Purpose

The User tag defines a person who has access to a trial database with the InForm system. A user's access to specific aspects of a trial are determined by the user's inclusion in the following definitions:

- **SiteGroup** (on page 201)—Gives the user access to a specific trial site.
- **RightsGroup** (on page 176)—Gives the user access to a set of rights to perform specific activities.
- **QueryGroup** (on page 162)—Gives a user who has been assigned the right to close queries the additional right to close queries initiated by another member of the same Query Group.
- **SignatureGroup** (on page 192)—Gives a user the right to sign documents requiring signature. To sign a site's documents, a user must be in a SignatureGroup and the appropriate SiteGroup.
- **ManagerGroup** (see "**ManagerUserGroup**" on page 155)—Gives the user access to CRA list for report viewing.

A user can also be included in the definition of a **Sponsor** (on page 202) or **Site** (on page 198). Inclusion in either of those definitions indicates that the user is a contact at the Sponsor or site location.

Note that if you attempt to install information about an existing, active user, including the user's password, InForm application makes the user inactive as a security precaution.

### Syntax

```
<USER
 USERNAME="name"
 USERTYPE="SYSTEM|SITE|SPONSOR"
 [FIRSTNAME="name"]
 [LASTNAME="name"]
 [DISPLAYNAME="name"]
 [DESCRIPTION="text"]
 [TITLE="name"]
 [ADDRESS="addr1"]
 [ADDRESS2="addr2"]
 [CITY="name"]
 [STATE="name"]
 [PROVINCE="name"]
 [ZIPCODE="code"]
 [POSTCODE="code"]
 [COUNTRY="name"]
 [PHONE="num"]
 [ALTPHONE="num"]
 [FAX="num"]
 [EMAIL="addr"]
 [BEEPER="num"]
 HOMESCREENURL="url"
 [IMAGEFILE="file"]
 [IMAGETYPE="GIF|JPEG|TEXT"]
```

```
[LANGUAGE="name"]
[ACTIVESTATE="true | false"]
[DELETESTATE="true | false"]
[PASSWORD="name"]
```

```
[USERDATEFORMAT="MONTH_DAY_YEAR | DAY_MONTH_YEAR | YEAR_MONTH_D
AY"]/>
```

### Attributes

**USERNAME="name"** Name that identifies the user in the database. Required.

**USERTYPE="SYSTEM | SITE | SPONSOR"**

Type of user. Required:

- **SYSTEM**—User with specialized system capabilities. For example, when the InForm application generates a query automatically, the user name assigned as the query originator is "autoquery." The autoquery user is a system user.
- **SITE**—User associated with a site.
- **SPONSOR**—User associated with a sponsor.

**FIRSTNAME="name"** First name of the user. Optional.

**LASTNAME="name"**

Last name of the user. Optional.

**DISPLAYNAME="name"**

User name as displayed on the navigation pane in the InForm system. Maximum length is 63 characters; shorter strings are recommended. Optional.

**DESCRIPTION="text"**

Description of the user; for example, user's role in the trial. Optional.

**TITLE="name"**

Title of the user. Optional. This also appears on the navigation pane in the InForm system along with the user's name.

**ADDRESS="addr1"**

First line of the user's address. Optional.

**ADDRESS2="addr2"**

Second line of the user's address. Optional.

**CITY="name"**

City in which the user's address is located. Optional.

**STATE="name"**

State in which the user's address is located. (Not for use with Province, below.) Optional.

**PROVINCE="name"**

Province in which the user's address is located. (Not for use with State, above.) Optional.

**ZIPCODE="code"**

User's Zip code. (Not for use with Postcode, below.) Optional.

**POSTCODE="code"**

User's postal code. (Not for use with Zipcode, above.) Optional.

**COUNTRY="name"**

Country in which the user's address is located. Optional.

**PHONE="num"**

User's telephone number. Optional.

**ALTPHONE="num"**

User's alternate telephone number. Optional.

**FAX="num"**

User's fax number. Optional.

**EMAIL="addr"**

User's e-mail address. Optional.

**BEEPER="num"**

User's beeper number. Optional.

**HOMESCREENURL="url"**

Local or external URL identifying the initial screen that appears when a user logs in to the InForm system. The address must include the http:// prefix and identify the server on which the file is located by name or IP address.

**IMAGEFILE="file"**

Name of the image file that appears on the navigation panel in the InForm system above the user's name. Optional.

**IMAGETYPE="GIF|JPEG|TEXT"**

Type of image file: GIF, JPEG, or TEXT. Required if you specified a filename for Imagefile.

**LANGUAGE="name"**

User's preferred language. Optional. English is the default.

**ACTIVESTATE="true|false"**

Indicates whether the user is active: true or false. Optional. true is the default.

**DELETESTATE="true | false"**

Indicates whether the user has been terminated: true or false. Optional. false is the default. Terminated users remain in the database.

**PASSWORD="name"**

User's password. Optional.

**USERDATEFORMAT="MONTH\_DAY\_YEAR | DAY\_MONTH\_YEAR | YEAR\_MONTH\_DAY"**

Desired date format for viewable InForm pages for this particular user.

## Example

The following example adds a user named CRC to the database.

```
<USER USERNAME="crc"
 USERTYPE="SITE"
 FIRSTNAME="Clinical"
 LASTNAME="Research Coordinator"
 DISPLAYNAME="CRC"
 HOMESCREENURL="/inform1/custom/HomeDefault.html"
 IMAGEFILE="..\Resources\UserPics\StudyCoordinator75.gif"
 IMAGETYPE="GIF"
 ACTIVESTATE="false"
 DELETESTATE="false"
 PASSWORD="gcp"/>
 USERDATEFORMAT="MMM/DD/YYYY"
```

## UserImage

### Purpose

The UserImage tag enables you to update the user image for an existing user. When you use the UserImage tag, all other existing user information remains the same.

**Note:** To update other individual user attributes, use the InForm application administration user interface or use the MedML Installer utility. When you update a *User* (on page 227) definition with the MedML Installer utility, you must repeat the entire user information set; you cannot update single attributes other than the user image individually.

## Syntax

```
<USERIMAGE
 USERNAME="name"
 [IMAGEFILE="file"]
 [IMAGETYPE="GIF|JPEG|TEXT"]
 [LANGUAGE="name"]/>
```

### Attributes

**USERNAME="name"** Name that identifies the user in the database. Required.

**IMAGEFILE="file"**

Name of the image file that appears on the InForm navigation panel above the user's name. Optional; if you do not specify a filename, the MedML Installer utility removes the current image file if one is defined for the user.

**IMAGETYPE="GIF|JPEG|TEXT"**

Type of image file: GIF, JPEG, or TEXT. Required if you specified a filename for Imagefile.

**LANGUAGE="name"**

Language of the user image file. Optional; English is the default.

## Example

The following example illustrates the use of the UserImage tag to add the image in the xena.jpg file to the sm user.

```
<USERIMAGE USERNAME="sm"
 IMAGEFILE="..\Resources\UserPics\xena.jpg"
 IMAGETYPE="GIF"/>
```

## Userref

### Purpose

The Userref tag enables you to include previously defined *Users* (see "*User*" on page 227) in the definition of any of the following components:

- *QueryGroup* (on page 162)
- *RightsGroup* (on page 176)
- *SignatureGroup* (on page 192)
- *SiteGroup* (on page 201)

A Userref appears only as the child of an component in which it is included; it is not submitted as a stand-alone component.

Note that the only way to remove a user from a group is through the Admin function of the InForm application. You cannot remove a user by running the MedML Installer utility.

### Syntax

```
<USERREF
 USERNAME="name"/>
```

### Attributes

**USERNAME="name"**

Name of the user to include in an component, as specified in the Username attribute of the *User* (on page 227) definition.

### Example

The following example illustrates the inclusion of the users Homer and Marge in the definition of a *QueryGroup* (on page 162) called SiteXQueries.

```
<QUERYGROUP GROUPNAME="SiteXQueries">
 <USERREF USERNAME="Homer"/>
 <USERREF USERNAME="Marge"/>
</QUERYGROUP>
```

## VerbatimType

### Purpose

The VERBATIMTYPE tag identifies a specific type of verbatim defined in the dictionary. VERBATIMTYPE corresponds to item type in the Central Coding application.

### Syntax

```
<VERBATIMTYPE
```



```
NAME="name"/>
```

## Attributes

**NAME="name"** Type of verbatim item. Optional. The following verbatim types are defined for the dictionaries supported by the Central Coding application:

- AE—Adverse event
- DISEASE—Disease
- LABDATA—Lab data
- MEDPROD—Medical product

AE, DISEASE, and LABDATA are valid verbatim types for the MedDRA dictionary. MEDPROD is a valid verbatim type for the WHODD dictionary.

## Example

```
<DICTIONARY TYPE="WHODD" VERSION="05Q4" CULTURE="en-US">
 <CODETARGET NAME="ATC 1.CODE"/>
 <CODETARGET NAME="ATC 1.TERM"/>
 <CODETARGET NAME="ATC 2.CODE"/>
 <CODETARGET NAME="ATC 2.TERM"/>
 <CODETARGET NAME="ATC 3.CODE"/>
 <CODETARGET NAME="ATC 3.TERM"/>
 <CODETARGET NAME="ATC 4.CODE"/>
 <CODETARGET NAME="ATC 4.TERM"/>
 <CODETARGET NAME="Preferred Name.CODE"/>
 <CODETARGET NAME="Preferred Name.TERM"/>
 <CODETARGET NAME="Ingredients.AddInfo"/>
 <CODETARGET NAME="Trade Name.CODE"/>
 <CODETARGET NAME="Trade Name.TERM"/>
 <CODETARGET NAME="Medicinal Product.CODE"/>
 <CODETARGET NAME="Medicinal Product.TERM"/>
 <CODETARGET NAME="Name Specifier.AddInfo"/>
 <CODETARGET NAME="Country of Sale.AddInfo"/>
 <CODETARGET NAME="MA Holder.AddInfo"/>
 <CODETARGET NAME="MA Holder Country.AddInfo"/>
 <CODETARGET NAME="Company.AddInfo"/>
 <CODETARGET NAME="Company Country.AddInfo"/>
 <CODETARGET NAME="ICH Med Prod ID.AddInfo"/>
 <CODETARGET NAME="Sequence Number 3.AddInfo"/>
 <CODETARGET NAME="Sequence Number 4.AddInfo"/>
 <CODETARGET NAME="MA Number.AddInfo"/>
 <CODETARGET NAME="MA Date.AddInfo"/>
 <CODETARGET NAME="MA Withdrawal Date.AddInfo"/>
 <CODETARGET NAME="Product Type.AddInfo"/>
 <CODETARGET NAME="Product Group.AddInfo"/>
 <CODETARGET NAME="Pharmaceutical Product.AddInfo"/>
 <CONTEXTITEM NAME="Route Of Administration"/>
 <CONTEXTITEM NAME="Indication"/>
 <VERBATIMTYPE NAME="MEDPROD" />
```

</DICTIONARY>

## Scripting object reference

### Conversion object

The InForm application enables you to define form components that you specify as units. You can define unit form components with reference to other units, so that data can be displayed online as one unit and stored in the database, after conversion, as the other. For example, you can design a form so that a CRC can choose whether to enter a weight in ounces or pounds and specify that the weight is always stored in the database in ounces. The unit in which data is stored in the database after conversion is called the base unit, and the data value, after conversion to the base unit, is called the normalized value.

To perform conversions between the entered and base unit values, you create rules with a rule type of Conversion. The scripts for these rules can use the Data object described in this topic.

### Data object

The Data object enables you to specify how to perform a conversion between the entered value of a data item and the normalized value stored in the database after conversion. A conversion rule is attached in an .xml file to a unit definition, which includes the REFNAME of the rule. For an example, see the *Unit* (on page 220) topic in the MedML Installer utility online Help.

The Data object has the following properties:

Property	Type	Purpose
Result	FLOAT	Holds the result of the conversion
BaseValue	NUMERIC	Contains the value to be converted

### Example

The following conversion rule script converts from centimeters to millimeters:

```
Data.Result=Data.BaseValue*10
```

## Execution plan objects and methods

Execution plans are associated with events, and an execution plan runs when its event fires. The InForm application supports execution plans that do either of the following:

- Send email
- Log a message to the NT log

A script that is part of the definition of an execution plan can use the objects and methods described in this topic.

**Note:** You can define execution plans only by using MedML and the MedML Installer utility.

## Global object

The Global object enables you to retrieve information about the current site, the user whose action caused the execution plan to run, or the user designated as the primary contact at the current site. Additionally, the Global object enables you to pass a message generated by a rule to an email message or to the NT log. The Global object has the following properties, which are Get only.

Property	Purpose
Site	Holds the current properties of the Site object, which contains site data.
SiteUser	Holds the current properties of the User object, which contains user data, for the user who is specified as the primary contact for the current site.
User	Holds the current properties of the User object, which contains user data, for the user whose action caused the execution plan to fire.
ContextString	Holds the message received from the Message property of the Patient or Result object. ContextString is a string property.

## Global methods

Method	Purpose
<i>EMailToGroup</i> (see " <i>EMailToGroup(GroupID, Subject,Msg)</i> " on page 239)	Sends email to the members of a specified query group, signature group, or user manager group defined in the trial database.
<i>EMailToInFormUser</i> (see " <i>EMailToInFormUser InFormUser,szSubject,szMsg</i> " on page 239)	Sends email to a specified InForm user.
<i>EMailToInternetUser</i> (see " <i>EMailToInternetUser EmailAddress,szSubject,szMsg</i> " on page 240)	Sends email to any internet email alias.
EMailToUser	For internal use only.
<i>GetNamedValue</i> (see " <i>GetNamedValue(Name)</i> " on page 240)	Returns the data value associated with a name specified in the AddNamedValue method on the Patient or Result object.
<i>LogMessage</i> (see " <i>LogMessage Message</i> " on page 241)	Logs the specified message to the NT log.
<i>OutputDebug</i> (see " <i>OutputDebug(PrintString, Variable)</i> " on page 241)	Displays the value of a specified variable in the InForm Architect application Output window as a test script runs.

## Site object

The Site object enables you to retrieve information about the current site. The Site object has the following properties, which are Get only:

Property	Type	Purpose
ContactID	DWORD	Holds the database Id of the InForm software user indicated as the primary contact for the site
SiteName	string	Holds the name of the site
Address	string	Holds the site address
Phone	string	Holds the site phone number
AltPhone	string	Holds the alternate site phone number
Beeper	string	Holds the beeper number at which to contact the site
Email	string	Holds the email address at which to contact the site
Fax	string	Holds the fax number at which to contact the site
TimeZone	string	Holds the site time zone

## Site method

Method	Purpose
<b>SetSite</b> (see " <b>SetSite(SiteID)</b> " on page 241)	Loads the Site object from cache with data from the site with the specified site Id. Oracle use only.

## User object

The User object enables you to retrieve information about the current user. The User object has the following properties, which are Get only:

Property	Type	Purpose
FirstName	string	Holds the user's first name
LastName	string	Holds the user's last name
Address	string	Holds the user's address
Phone	string	Holds the user's phone number
AltPhone	string	Holds the alternate user phone number
Beeper	string	Holds the beeper number at which to contact the user
Email	string	Holds the email address at which to contact the user
Fax	string	Holds the fax number at which to contact the user
UserName	string	Holds the user account name used to log in to the InForm application

## User method

Method	Purpose
<b><i>SetUser</i></b> (see " <b><i>SetUser(UserID)</i></b> " on page 241)	Loads the User object from cache with data from the user with the specified user Id. Oracle use only.

## EMailToGroup(GroupID,Subject,Msg)

EMailToGroup is for internal use only.

The EMailToGroup method sends email to the members of a specified query group, signature group, or user manager group defined in the trial database.

### Arguments

**GroupID** Internally assigned database Id of the query group, signature group, or user manager group to receive the email.

**Subject** String containing the subject of the email.

**Msg** String containing the text of the email message.

**Note:** The GroupID argument identifies the recipient of the email. The following registry entry specifies the sender (the From address):  
 HKLM/SOFTWARE/PhaseForward/InForm/PFMngrExecutionPlan. The entry is a string value named "FromAddress". The default is "nobody@pf.com".

## EMailToInFormUser "InFormUser,szSubject,szMsg"

The EMailToInFormUser method sends email to a specified InForm user.

### Arguments

**InFormUser** Internally assigned database Id of the InForm user to receive the email.

**Subject**

String containing the subject of the email.

**Msg**

String containing the text of the email message.

**Note:** The InFormUser argument identifies the recipient of the email. The following registry entry specifies the sender (the From address):  
 HKLM/SOFTWARE/Oracle/InForm/PFMngrExecutionPlan. The entry is a string value named "FromAddress". The default is "nobody@pf.com".

### Example

Global.EMailToInFormUser "sm", "CRFUnLock", "Sending Inform User email when Unlockl-CRF"

## EMailToInternetUser "EmailAddress,szSubject,szMsg"

The EMailToInternetUser method sends email to any internet email alias.

### Arguments

**EmailAddress** The internet email address of the InForm user to receive the email.

**Subject**

String containing the subject of the email.

**Msg**

String containing the text of the email message.

**Note:** The EmailAddress argument identifies the recipient of the email. The following registry entry specifies the sender (the From address):  
**HKLM/SOFTWARE/Oracle/InForm/PFMngrExecutionPlan.** The entry is a string value named "FromAddress". The default is "nobody@Oracle.com".

### Example

```
Global.EMailToInternetUser "user@Oracle.com", "CRFUnlock", "Sending External User email
when Unlock-CRF"
```

## GetNamedValue(Name)

The GetNamedValue method returns the data value associated with a name specified in the AddNamedValue method on the Patient or Result object.

### Argument

**Name** String containing the name passed by the AddNamedValue method.

### Example

```
aeserval_m = Global.GetNamedValue("aeserval")
if aeserval_m = "Y" then
 ptinit_m = Global.GetNamedValue("ptinit")
 aedate_m = Global.GetNamedValue("aedate")
 aeevent_m = Global.GetNamedValue("aeevent")
 site_m = Global.GetNamedValue("sitenm")
 emailtxt="An SAE has occurred at Site: "&site_m&" for Patient: "&ptinit_m &" on AE Date:
"&aedate_m&". The event is : "&aeevent_m&".
 call Global.EmailtoInternetUser("test@pf.com", "SAE Occurrence",emailtxt)
end if
```



## LogMessage "Message"

The LogMessage method logs the specified message to the NT log.

### Argument

**Message** String containing the text of the message to log to the NT log.

### Example

```
Global.LogMessage "Query generated"
```

## OutputDebug(PrintString, Variable)

The OutputDebug method displays the value of a specified variable in the InForm Architect application Output window as a test script runs.

It is not necessary to remove this before running the InForm application, as this method is non-operational in the InForm system.

### Arguments

**PrintString** String that identifies the value to trace.

**Variable**

Name of the variable for which to display the current value.

### Example

```
UnitFilled = Patient.ItemHasBeenNormalizedEx("0.Visit1.DEM.DEM.0.HEIGHT",0,0)
Patient.OutputDebug "TestUnit", UnitFilled
```

## SetSite(SiteID)

SetSite(SiteID) is for internal use only. The SetSite method loads the Site object from cache with data from the site with the specified site Id.

### Argument

**SiteID** Database Id for the site you want to access by using the Site object.

## SetUser(UserID)

SetUser(UserID) is for internal use only. The SetUser method loads the User object from cache with data from the user with the specified user Id.

### Argument

**UserID** Database Id for the user you want to access by using the User object.

## Randomization objects and methods

When you implement randomization in a trial, you must perform several configuration steps. One of these is to create a script that generates a drug kit number to assign to each patient as the patient is enrolled. This randomization script can use the objects and methods that are available for rules and calculations and also use the Randomization object and method described in this topic.

By using MedML and the MedML Installer utility, you attach the randomization script to a calculated control on the form where you want to generate the randomization sequence number and assign a drug kit. A predefined section definition called DRUGKITSECTION is provided in the *randomization.xml* file. For details, see the *Setting Up a Trial with InForm Architect and MedML Guide*.

**Randomization object** (on page 242)

**GetNextKit method** (on page 243)

**GetNextNumber method** (on page 245)

### Randomization object

The Randomization object enables you to assign the patient, site, randomization, and trial revision information that the GetNextKit method uses to determine the next drug kit number. The Randomization object has the following properties:

Property	Purpose
PatientID	Holds the ID of the current patient.
SiteID	Holds the ID of the current site. The site ID is an internally assigned value by which the site is known in the database.
Type	<p>Holds the randomization type:</p> <ul style="list-style-type: none"> <li>• 1 (Simple Central)—The trial uses one list of drug kits. Each new patient is assigned the next sequential drug kit number on the list.</li> <li>• 2 (Central Stratified)—The trial has multiple lists of drug kits. Each new patient is assigned to a drug kit list based on entered patient data. Then, the patient is assigned the next sequential drug kit number on that list.</li> <li>• 3 (Simple Site)—Each site has a different drug kit list. Each new patient is assigned the next sequential drug kit number on the list for the patient's site.</li> <li>• 4 (Stratified by Site)—Each site has multiple lists of drug kits. Each new patient is first assigned to the set of list for the patient's site. Then, the patient is assigned to one of the site's drug kit lists based on entered patient data. Finally, the patient is assigned the next sequential drug kit number on that list.</li> </ul>
Source	Randomization source list name, or stratification code.
Revision	Revision number of the trialversion.

## GetNextKit method (KitInfo)

The GetNextKit method gets the next sequence number from the randomization source specified in the Randomization.Source property. If the next number in the sequence has additional kit information associated with it, GetNextKit also returns the corresponding kit information.

### Argument

Kit information associated with the returned sequence number, if provided.

### Examples

The following example illustrates a randomization rule for a Simple Central (Type 1) randomization scheme.

```
Randomization.SiteID = Patient.GetSiteID
Randomization.Type = 1
Randomization.Source = "SimpleCentral"
Randomization.PatientID = Patient.GetID
Randomization.Revision = 0
SeqNum = Randomization.GetNextKit(KitInfo)
Result.Result= SeqNum + " / " + KitInfo
```

The following example illustrates a randomization rule for a Central Stratified (Type 2) randomization scheme. In this example, the stratification is based on the patient's weight, as entered in the Demographics form in Visit 1. Based on the patient's weight, the InForm application gets the next sequence number from either the CS\_WT150 or the CS\_WT275 randomization source list.

```
Function GetRndSourceList()
wt = Patient.GetValue("0.Visit1.DEM.DEM.0.WEIGHT", "", 0,0,0)
IF (wt > 90 AND wt < 150) THEN
GetRndSourceList = "CS_WT150"
ELSEIF (wt > 150 AND wt < 275) THEN
GetRndSourceList = "CS_WT275"
ELSE
GetRndSourceList = ""
END IF
End Function
```

```
'set properties
Randomization.SiteID = Patient.GetSiteID
Randomization.Type = 2
Randomization.Source = GetRndSourceList
Randomization.PatientID = Patient.GetID
Randomization.Revision = 0
'randomize the patient and return result
SeqNum = Randomization.GetNextKit(KitInfo)
Result.Result= SeqNum + " / " + KitInfo
```

The following example illustrates a randomization rule for a Simple Site (Type 3) randomization scheme. In this example, the patient's site determines the randomization source list from which InForm application gets the next sequence number.

```
Function GetRndSourceList()
```

```

szSite = Patient.GetSiteName();
IF (szSite = "PhaseForward") THEN
 GetRndSourceList = "SS_PF"
ELSEIF (szSite = "Beth Israel") THEN
 GetRndSourceList = "SS_BID"
ELSE
 GetRndSourceList = ""
END IF
End Function

```

```

'set properties
Randomization.SiteID = Patient.GetSiteID
Randomization.Type = 3
Randomization.Source = GetRndSourceList
Randomization.PatientID = Patient.GetID
Randomization.Revision = 0
'run the randomization and return result
SeqNum = Randomization.GetNextKit(KitInfo)
Result.Result= SeqNum + " / " + KitInfo

```

The following example illustrates a randomization rule for a Stratified by Site (Type 4) randomization scheme. In this example, the patient's site and height determine the randomization source list from which the InForm application gets the next sequence number. This example uses the Oracle randomization source lists SR\_PF\_HT45 and SR\_PF\_HT75 and the Beth Israel randomization source lists SR\_BID\_HT45 and SR\_BID\_HT75.

```

Function GetRndSourceList()
szSite = Patient.GetSiteName();
ht = Patient.GetValue("0.Visit1.DEM.DEM.0.HEIGHT", "", 0,0,0)

SELECT CASE szSite
CASE "Oracle"
IF (ht > 30 AND ht < 45) THEN
 GetRndSourceList = "SR_PF_HT45"
ELSEIF (ht > 45 AND < 75) THEN
 GetRndSourceList = "SR_PF_HT75"
ELSE
 GetRndSourceList = ""
END IF

CASE "Beth Israel"
IF (ht > 30 AND ht < 45) THEN
 GetRndSourceList = "SR_BID_HT45"
ELSEIF (ht > 45 AND < 75) THEN
 GetRndSourceList = "SR_BID_HT75"
ELSE
 GetRndSourceList = ""
END IF

CASE ELSE
 GetRndSourceList = ""
END SELECT
End Function

```

```
'setup randomization object properties
Randomization.SiteID = Patient.GetSiteID
Randomization.Type = 4
Randomization.Source = GetRndSourceList
Randomization.PatientID = Patient.GetID
Randomization.Revision = 0
'randomize the patient and return result
SeqNum = Randomization.GetNextKit(KitInfo)
Result.Result= SeqNum + " / " + KitInfo
```

## GetNextNumber method (KitInfo)

The GetNextNumber method gets the next sequence number from the randomization source specified in the Randomization.Source property. This method differs from the GetNextKitNumber method in that the additional kit information associated with the sequence number is not returned.

## Argument

Kit information associated with the returned sequence number, if provided.

## Rule and calculation objects and methods

Scripts for form rules—scripts that perform edit checks on data entered into an item on a form—and calculations—scripts that compute the value of a data item on a form based on the value of one or more related items—can use the objects described in this topic.

**Note:** You can create form rule and calculation definitions either by using the rule definition user interface in InForm Architect application or by using MedML and the MedML Installer utility.

## Patient object

The Patient object enables you to obtain form values that pertain to a patient. The Patient object has the following properties, all of which are Get and Put properties:

Property	Type	Purpose
Result	VARIANT	Holds the result of a calculation
Message	BSTR	Passes a string to the ContextString property of the Global object used in a script for an execution plan. This property applies only to form rules
RulePassed	BOOL	Specifies whether an edit check passed or failed. If RulePassed is true (equal to 1), the edit check passes; if false (equal to 0), the edit check fails.
BaseValue	NUMERIC	Contains the value to be converted

## Patient methods

Scripts for form rules—scripts that perform edit checks on data entered into an item on a form—and calculations—scripts that compute the value of a data item on a form based on the value of one or more related items—can use many of the same methods. Except where noted, Patient methods are available for both form rules and calculations.

Method	Purpose
<i>AddNamedValue</i> (see " <i>AddNamedValue</i> "Name", "Value" on page 251)	Adds a Name-Value pair to the list of arguments that the GetNamedValue method can access.
<i>ClearValues</i> (see " <i>ClearValues()</i> " on page 252)	Clears the properties of the specified object.
<i>FormHasState</i> (see " <i>FormHasState(Visit, Form, State)</i> " on page 253)	Indicates whether a form is in a state that you specify. FormHasState returns a boolean value.
<i>FormHasStateEx</i> (see " <i>FormHasStateEx(Visit, VisitIndex, Form, State)</i> " on page 254)	Returns a boolean value indicating whether a form is in a specified state.
<i>FormHasStateRF</i> (see " <i>FormHasStateRF(Visit, VisitIndex, Form, FormIndex, State)</i> " on page 256)	Returns a boolean value indicating whether an instance of a repeating form is in a specified state.
<i>GetArgument</i> (on page 257)	Returns the value of the specified rule argument.
<i>GetArgumentCount</i> (see " <i>GetArgumentCount()</i> " on page 257)	Returns the number of arguments used in the rule context.
<i>GetAssociationCount</i> (see " <i>GetAssociationCount(RefNamePath, VisitIndex, FormIndex)</i> " on page 258)	Returns the number of repeating form instances that have been associated with a specific instance of a form within a visit.
<i>GetAssociationValue</i> (see " <i>GetAssociationValue(nAssociations)</i> " on page 259)	Iterates over the associations of a form until it reaches the count returned in the latest call to the GetAssociationCount method. When it reaches the association represented by the GetAssociationCount value, GetAssociationValue returns an array containing RefNames and indices for the specified associated form instance.
<i>GetCurPath</i> (see " <i>GetCurPath()</i> " on page 259)	Returns the path of the context in which the rule is running.

<b><i>GetCurrentFormIndex</i></b> (see " <b><i>GetCurrentFormIndex()</i></b> " on page 260)	Returns a number containing the index of the current instance of a repeating form.
<b><i>GetCurrentISRowNum</i></b> (see " <b><i>GetCurrentISRowNum()</i></b> )" on page 260)	Returns a number containing the index of the current row or 0 if it is a new row. (Always returns 0 in the InForm Architect application.)
<b><i>GetCurrentVisitIndex</i></b> (see " <b><i>GetCurrentVisitIndex()</i></b> " on page 261)	Returns a number containing the index of the current visit.
<b><i>GetDefaultValue</i></b> (see " <b><i>GetDefaultValue(RefNamePath)</i></b> " on page 262)	Is the same as GetValue, with default parameters.
<b><i>GetEnteredValue</i></b> (see " <b><i>GetEnteredValue(REFNAMEPath,AttributeName,Index,ItemSetIndex,VisitIndex)</i></b> " on page 264)	Returns the value of the data entered in the specified form component.
<b><i>GetEnteredValueRF</i></b> (see " <b><i>GetEnteredValueRF(REFNAMEPath,AttributeName,ItemsetIndex,VisitIndex,FormIndex)</i></b> " on page 269)	Returns the value of the data entered in the specified form component in a specific instance of a repeating form.
<b><i>GetFormCount</i></b> (see " <b><i>GetFormCount(Visit,VisitIndex,Form)</i></b> " on page 272)	Returns the returns the number of repeating form instances that have been created for a specified visit.
<b><i>GetID</i></b> (see " <b><i>GetID()</i></b> " on page 273)	Returns the internal database Id of the current patient.
<b><i>GetItemsetCount</i></b> (../cookerhelp/meth_getitemsetcount.htm)	Returns the number of rows currently entered in an itemset. Note: Do not use this as a means to sequentially number or identify a row. See GetItemSetRowID, below.
<b><i>GetItemsetRowID</i></b> (../cookerhelp/meth_getitemsetrowid.htm)	Returns the uniquely identifying row ID of an itemset row.
<b><i>GetItemsetCount</i></b> (see " <b><i>GetItemsetCount(REFNAMEPath,VisitIndex)</i></b> " on page 273)	Returns the number of rows currently entered in an itemset.

<i>GetItemsetCountRF</i> (see " <i>GetItemsetCountRF(REFNAMEPath, VisitIndex, FormIndex)</i> " on page 275)	Returns the number of rows currently entered in an itemset in a specific instance of a repeating form.
<i>GetPatientNumber</i> (see " <i>GetPatientNumber()</i> " on page 277)	Returns the number assigned to the current patient.
<i>GetRefName</i> (see " <i>GetRefName(ObjType)</i> " on page 277)	Returns a string containing the RefName of the specified object type in the context in which the rule is run.
<i>GetRefPath</i> (see " <i>GetRefPath(ObjType)</i> " on page 277)	Returns a string containing the RefName path of the specified object type in the context in which the rule is run.
<i>GetScreeningNumber</i> (see " <i>GetScreeningNumber()</i> " on page 277)	Returns a string containing the current patient's screening number.
<i>GetServerTime</i> (see " <i>GetServerTime()</i> " on page 278)	Returns a string containing the datetime value of the current server as GMT.
<i>GetSiteID</i> (see " <i>GetSiteID()</i> " on page 278)	Returns the internal Id of the current patient's site.
<i>GetSiteMnemonic</i> (see " <i>GetSiteMnemonic()</i> " on page 278)	Returns a string containing the site mnemonic for the current patient's site.
<i>GetSiteName</i> (see " <i>GetSiteName()</i> " on page 278)	Returns the name of the current patient's site.
<i>GetSiteTime</i> (see " <i>GetSiteTime()</i> " on page 278)	Returns the datetime value of the server in the current site's time zone.
<i>GetValue</i> (see " <i>GetValue(REFNAMEPath, AttributeName, Index, ItemsetIndex, VisitIndex)</i> " on page 279)	Returns the normalized value of the data entered in the specified form component.
<i>GetValueRF</i> (see " <i>GetValueRF(REFNAMEPath, AttributeName, Index, ItemsetIndex, VisitIndex, FormIndex)</i> " on page 281)	Returns the normalized value of the data entered in the specified form component in a specific instance of a repeating form.



<i>GetVisitCount</i> (see " <i>GetVisitCount(VisitRefName)</i> " on page 285)	Returns the number of visits for the visit specified in VisitRefName.
<i>IsCBSelected</i> (see " <i>IsCBSelected(RefNamePath,ItemSetIndex,VisitIndex)</i> " on page 285)	Checks to see if a SimpleControl in a CheckBoxControl is checked and returns a Boolean containing the state of the check box.
<i>IsCBSelectedRF</i> (see " <i>IsCBSelectedRF(RefNamePath,ItemsetIndex,VisitIndex,FormIndex)</i> " on page 287)	In a specific instance of a repeating form, checks to see if a SimpleControl in a CheckBoxControl is checked and returns a Boolean containing the state of the check box.
<i>IsItemSetDeleted</i> (see " <i>IsItemSetDeleted(REFNAMEPath,VisitIndex,ItemSetIndex)</i> " on page 289)	Indicates whether an item in an itemset is deleted
<i>IsItemSetDeletedRF</i> (see " <i>IsItemSetDeletedRF(REFNAMEPath,VisitIndex,FormIndex,ItemsetIndex)</i> " on page 290)	Indicates whether an item in an itemset is deleted in a specific instance of a repeating form.
<i>IsValidItemPath</i> (see " <i>IsValidItemPath(RefNamePath,VisitIndex)</i> " on page 291)	Confirms whether a RefName path is valid, by checking the database for the existence of the specified path.
<i>IsValidItemPathRF</i> (see " <i>IsValidItemPathRF(RefNamePath,Visit,Index,FormIndex)</i> " on page 293)	In a specific instance of a repeating form, confirms whether a RefName path is valid, by checking the database for the existence of the specified path.
<i>ItemHasBeenNormalizedEx</i> (see " <i>ItemHasBeenNormalizedEx(REFNAMEPath,ItemsetIndex,VisitIndex)</i> " on page 295)	Indicates whether the units associated with the referenced data item have been selected, by determining whether the data has been normalized.
<i>ItemHasBeenNormalizedRF</i> (see " <i>ItemHasBeenNormalizedRF(REFNAMEPath,ItemsetIndex,VisitIndex,FormIndex)</i> " on page 296)	In a specific instance of a repeating form, indicates whether the units associated with the referenced data item have been selected, by determining whether the data has been normalized.
<i>LoadCRB</i> (see " <i>LoadCRB()</i> " on page 298)	Loads the patient Case Book into memory.

<i>OutputDebug</i> (see " <i>OutputDebug(PrintString, Variable)</i> " on page 241)	Displays the value of a specified variable in the InForm Architect application Output window as a test script runs.
<i>RemoveEmptyDynamicVisit</i> (on page 299)	Removes a dynamic visit that a patient has been scheduled into, if no data has been entered into any of the visit's forms.
<i>RemoveNotStartedDynamicForm</i> (on page 299)	Removes a dynamic form from a patient's Case Book, provided the form has not been started.
<i>SetDynamicVisitStart</i> (on page 300)	Schedules a patient into a dynamic visit.
<i>SetEnteredValue</i> (see " <i>SetEnteredValue(REFNAMEPath, AttributeName, Index, ItemsetIndex, VisitIndex, Value)</i> " on page 301)	Uses the value of the data entered in the specified form component.
<i>SetEnteredValueRF</i> (see " <i>SetEnteredValueRF(REFNAMEPath, AttributeName, Index, ItemsetIndex, VisitIndex, FormIndex, Value)</i> " on page 304)	In a specific instance of a repeating form, uses the value of the data entered in the specified form component.
<i>SetNewValue</i> (see " <i>SetNewValue(REFNAMEPath, AttributeName, Index, ItemsetIndex, VisitIndex, Value)</i> " on page 307)	For internal Oracle use only.
<i>SetNewValueRF</i> (see " <i>SetNewValueRF(REFNAMEPath, AttributeName, Index, ItemsetIndex, VisitIndex, FormIndex, Value)</i> " on page 309)	For internal Oracle use only.
<i>SetValue</i> (see " <i>SetValue(REFNAMEPath, AttributeName, Index, ItemsetIndex, VisitIndex, Value)</i> " on page 312)	Uses the normalized value of the data entered in the specified form component.
<i>SetValueRF</i> (see " <i>SetValueRF(REFNAMEPath, AttributeName, Index, ItemsetIndex, VisitIndex, FormIndex, Value)</i> " on page 314)	In a specific instance of a repeating form, uses the normalized value of the data entered in the specified form component.

<i>ShowDynamicForm</i> (on page 317)	Inserts a dynamic form into a visit based on the outcome of a calculation on another form.
--------------------------------------	--------------------------------------------------------------------------------------------

## Result object

The Result object enables you to get or set the result of an edit check generated by a form rule or the data item value generated by a calculation. The properties of the Result object are identical to the properties of the Patient object.

### Result methods

The methods available for use with the Result object are the same methods as are available for the **Patient object**.

## AddNamedValue"Name","Value"

The AddNamedValue method adds a Name-Value pair to the list of arguments that the GetNamedValue method can access. GetNamedValue is a method on the Global object used in the definition of an execution plan. Additionally, you can use this method to generate dynamic query text that overrides the default text specified in the Event definition associated with the rule.

The following predefined Name-Value pairs are available to use for dynamic query text generation:

- QUERYTEXT—Use with a string Value specifying the text of a query.
- QUERYSTATE—Use with the Values "Candidate" or "Open" to specify the state of a dynamically generated query. These Values not case sensitive.

## Arguments

**Name** String representing the name of the item. This is a simple string, not the item's REFNAME.

### Value

Value to be added.

### Example 1

```
'Check if ae is serious
'-----
AESer = "0.CommonCRF.AVR.AE.AE.SAE"
AESerVal_g=Patient.GetValue(AESer,"",0,0,0)
if AESerVal_g = "Y" then
sitenm_g=Patient.GetSiteName()
ptinit_g=cstr(Patient.GetValue("0.screen.screen.screen.0.patientinitials","",0,0,0))
aedate_g=Patient.GetValue("0.CommonCRF.AVR.AE.AE.STARTDT","",0,0,0)
aeevent_g=Patient.GetValue("0.CommonCRF.AVR.AE.AE.AEDESC","",0,0,0)
patient.AddNamedValue "sitenm", sitenm_g
patient.AddNamedValue "ptinit", ptinit_g
patient.AddNamedValue "aedate", aedate_g
patient.AddNamedValue "aeevent", aeevent_g
patient.AddNamedValue "aeserval", aeserval_g
End If
```

### Example 2

```
Result.RulePassed = 0
Patient.AddNamedValue "QUERYTEXT", "Wrist circumference is present but unit is missing;
please verify"
if (Patient.IsItemSetDeleted(Patient.GetCurPath(),0,0)) then
 Patient.AddNamedValue "QUERYSTATE", "Candidate"
else
 Patient.AddNamedValue "QUERYSTATE", "Open"
End if
```

### Example 3

The following script checks whether a systolic component of a blood pressure measurement is less than the diastolic component. If the rule fails, the rule calls the AddNamedValue method to issue a query whose text overrides the default query text specified in the Event definition for the rule:

```
Measure1 = Patient.GetValue(Patient.GetArgument("SysItem"), "", 0,0,0)
Measure2 = Patient.GetValue(Patient.GetArgument("DiasItem"), "", 0,0,0)

if (Measure1<Measure2) Then
 Result.RulePassed=0
 Patient.AddNamedValue "QUERYTEXT", "Systolic value must be higher than diastolic"

 Patient.AddNamedValue "QUERYSTATE", "Open"
Else
 Result.RulePassed=1
End If
```

### ClearValues()

The ClearValues method clears the properties and Name- Value pairs of the specified object.

## FormHasState(Visit,Form,State)

The FormHasState method indicates whether a form is in a state that you specify. FormHasState returns a Boolean value. Use FormHasState for regular, nonrepeating visits. To target a specific instance of an unscheduled, repeating visit, use *FormHasStateEx* (see "*FormHasStateEx(Visit, VisitIndex, Form, State)*" on page 254).

### Arguments

**Visit** REFNAME of the visit. Note that this is the REFNAME only of the visit, not the full REFNAMEPath.

#### Form

REFNAME of the form. Note that this is the REFNAME only of the form, not the full REFNAMEPath.

#### State

String specifying the state you want to check. The following states are valid:

- frozen—Indicates that the form is frozen
- hascomment—Indicates that the form has a comment at the form level
- hasdata—Indicates that the form has data
- locked—Indicates that the form is locked
- missing—Indicates that the form has missing data
- queries—Indicates that the form has queries
- sdvcomplete—Indicates that the form has been marked completely source verified
- sdvpartial—Indicates that the form has been partly source verified
- sdvready—Indicates that the form has been marked ready for source verification
- signed—Indicates that the form has been signed
- skipped—Indicates that the form has been marked not completed with a form-level comment
- started—indicates that the form was started with patient data, comments, or queries

### Example

For example, if an AE form indicates a ConMed was taken, verify the ConMed form has data. If Patient.FormHasState("CommonVisit", "ConMed", "missing") then

```
Result.RulePassed = 0
```

Else

```
Result.RulePassed = 1
```

## FormHasStateEx(Visit,VisitIndex,Form,State)

The FormHasStateEx method indicates whether a form is in a state that you specify. FormHasStateEx returns a Boolean value. FormHasStateEx is exactly the same as *FormHasState* (see "*FormHasState(Visit,Form,State)*" on page 253) except that FormHasStateEx includes the VisitIndex argument so that you can target a specific instance of an unscheduled visit.

### Arguments

**Visit** REFNAME of the visit. Note that this is the REFNAME only of the visit, not the full REFNAMEPath.

#### VisitIndex

Number that specifies an index into a repeating formset. A repeating formset is most often used for unscheduled visits in which the same data is recorded each time the visit occurs. Valid VisitIndex values are 0 or greater:

- 0—Indicates that the formset is not repeating or, if the formset is repeating, references the current formset.
- Any number greater than 0—Explicitly indexes the specified occurrence of the formset.

**Form** REFNAME of the form. Note that this is the REFNAME only of the form, not the full REFNAMEPath.

#### State

String specifying the state you want to check. The following states are valid:

- frozen—Indicates that the form is frozen
- hascomment—Indicates that the form has a comment at the form level
- hasdata—Indicates that the form has data
- locked—Indicates that the form is locked
- missing—Indicates that the form has missing data
- queries—Indicates that the form has queries
- sdvcomplete—Indicates that the form has been marked completely source verified
- sdvpartial—Indicates that the form has been partly source verified
- sdvready—Indicates that the form has been marked ready for source verification
- signed—Indicates that the form has been signed
- skipped—Indicates that the form has been marked not completed with a form-level comment
- started—indicates that the form was started with patient data, comments, or queries

## Example

For example, if an AE form in the second instance of an unscheduled visit indicates a concomitant medication was taken, verify the ConMed form in the same instance has data.

```
If Patient.FormHasStateEx("UnschVisit", "2", "ConMed", "missing") then
```

```
 Result.RulePassed = 0
```

```
Else
```

```
 Result.RulePassed = 1
```

## FormHasStateRF(Visit,VisitIndex,Form,FormIndex,State)

For repeating forms, the FormHasStateRF method indicates whether a form is in a state that you specify. FormHasStateRF returns a Boolean value. FormHasStateRF is exactly the same as *FormHasStateEx* (see "*FormHasStateEx(Visit,VisitIndex,Form,State)*" on page 254) except that FormHasStateRF:

- Includes the FormIndex argument so that you can target a specific instance of a repeating form
- Enables you to check for the deleted state

### Arguments

#### Visit

REFNAME of the visit. Note that this is the REFNAME only of the visit, not the full REFNAMEPath.

#### VisitIndex

Number that specifies an index into a repeating formset. A repeating formset is most often used for unscheduled visits in which the same data is recorded each time the visit occurs. Valid VisitIndex values are 0 or greater:

- 0—Indicates that the formset is not repeating or, if the formset is repeating, references the current formset.
- Any number greater than 0—Explicitly indexes the specified occurrence of the formset.

#### Form

REFNAME of the form. Note that this is the REFNAME only of the form, not the full REFNAMEPath.

#### FormIndex

Number that specifies an index into a repeating form.

- 0—References the current repeating form.
- 1—References a non-repeating form.
- Any number greater than 1—References the specified occurrence of the repeating form.

#### State

String specifying the state you want to check. The following states are valid:

- frozen—Indicates that the form is frozen
- hascomment—Indicates that the form has a comment at the form level
- hasdata—Indicates that the form has data
- locked—Indicates that the form is locked
- missing—Indicates that the form has missing data
- queries—Indicates that the form has queries
- sdvcomplete—Indicates that the form has been marked completely source verified



- `sdvpartial`—Indicates that the form has been partly source verified
- `sdvready`—Indicates that the form has been marked ready for source verification
- `signed`—Indicates that the form has been signed
- `skipped`—Indicates that the form has been marked not completed with a form-level comment
- `started`—indicates that the form was started with patient data, comments, or queries
- `deleted` (for instances of repeating forms)—indicates that the form instance has been deleted

## Example

The following example tests whether the text description of the adverse experience being entered in the `txtAEDiag` control already exists. It uses the `FormHasStateRF` method to determine whether the instance being tested has been deleted so that deleted instances can be excluded from the test.

```

patient.RulePassed = 1
Patient.AddNamedValue "QUERYTEXT", "This AE already exists"
path="0.CommonCRF.AE.sctAE.0.itmAEDiag.txtAEDiag"
isValid=patient.IsValidItemPathRF (path,0,0)
strVal=patient.GetValueRF(path,"",0,0,0,0)
intcurFormCount=patient.GetCurrentFormIndex()
intFormCount=patient.GetFormCount("CommonCRF",1,"AE")

for count = 1 to intFormCount
intIsDel=Patient.FormHasStateRF("CommonCRF",1,"AE",count,"deleted")
if (cint(count) <> cint(intcurFormCount)) and intIsDel=0 then
strVal2=patient.GetValueRF("0.CommonCRF.AE.sctAE.0.itmAEDiag.txtAEDiag","", 0, 0, 0,
count)
If Ucase(strVal) = Ucase(strVal2) then
patient.RulePassed = 0
exit for
End If
End If
Next

```

## GetArgument

The `GetArgument` method returns the value of the specified rule argument. Arguments are defined with `RULEARG` tags and associated with a rule and item in an `ATTACHRULESET` definition.

## Example

In this example, "Item1" is the name of an argument that has been defined for a rule and associated with the item being tested.

```
pulse = Patient.GetValue(Patient.GetArgument("Item1"), "", 0, 0, 0)
```

## GetArgumentCount()

The `GetArgumentCount` method returns the number of arguments used in the rule context.

## GetAssociationCount(RefNamePath,VisitIndex,FormIndex)

The GetAssociationCount method returns the number of repeating form instances that have been associated with a specific instance of a form within a visit.

### Arguments

**RefNamePath** String that identifies the form to get. Each RefName in the string is the RefName specified in the .xml file that contains the definition of the form component. The RefNamePath must be fully qualified; only the patient component of the path defaults to the current patient. The RefNamePath argument is in the following format: *(0.Visit.Form)*

- 0—Current patient.
- *Visit*—RefName of the visit, as specified in the .xml file that contains the visit definition.
- *Form*—RefName of the form, as specified in the .xml file that contains the form definition.

**VisitIndex** Number that specifies an index into a visit instance within the visit specified in the FormRefNamePath. A repeating formset is most often used for unscheduled visits in which the same data is recorded each time the visit occurs. Valid VisitIndex values are 0 or greater:

- 0—Indicates that the formset is not repeating or, if the formset is repeating, references the current formset.
- Any number greater than 0—Explicitly indexes the specified occurrence of the formset.

**FormIndex** Number that specifies an index into a repeating form. Form indexes start with 1 for the first instance of the form.

### Example

The following example uses the value returned by GetAssociationCount to set the number of times to test for missing data in a group of repeating form instances associated with the current form. GetAssociationCount works in conjunction with the **GetAssociationValue** (see "**GetAssociationValue(nAssociations)**" on page 259) method.

```
patient.RulePassed = 1
Patient.AddNamedValue "QUERYTEXT", "At least one of the Associated forms is missing data"
nAssociations = Patient.GetAssociationCount(Patient.GetCurPath(), Patient.GetCurrentVisitIndex(),
Patient.GetCurrentFormIndex())

for count = 1 to nAssociations
 Assoc = Patient.GetAssociationValue(count)
 AssocVisit = Assoc(0)
 AssocForm = Assoc(1)
 AssocVisitIndex = Assoc(2)
 AssocFormIndex = Assoc(3)
 'Do something with the association. Example: see if data missing
 if(Patient.FormHasStateRF(AssocVisit, AssocVisitIndex , AssocForm, AssocFormIndex, "missing"))
 then
 patient.RulePassed = 0
 end if
Next
```

## GetAssociationValue(nAssociations)

The GetAssociationValue method works in conjunction with the *GetAssociationCount* (see "*GetAssociationCount(RefNamePath, VisitIndex, FormIndex)*" on page 258) method to return information about a specific associated form instance. GetAssociationValue iterates over the associations of a form until it reaches the count returned in the latest call to the GetAssociationCount method. When it reaches the association represented by the GetAssociationCount value, GetAssociationValue returns an array containing the following values for the specified associated form instance:

- RefName of the visit
- RefName of the form
- VisitIndex
- FormIndex

## Arguments

**nAssociations** Number of instances of the associated form to iterate over, starting with 1 and including the specified number.

## Example

This example illustrates using the GetAssociationValue method to check whether an instance is signed.

```
nAssociations = Patient.GetAssociationCount(Patient .GetCurPath(), Patient.
GetCurrentVisitIndex(), Patient. GetCurrentFormIndex())
```

```
for count = 1 to nAssociations
 Assoc = Patient.GetAssociationValue(count)
 AssocVisit = Assoc(0)
 AssocForm = Assoc(1)
 AssocVisitIndex = Assoc(2)
 AssocFormIndex = Assoc(3)
 ' Do something with the association. Example: see if signed
 if(Patient.FormHasStateRF(AssocVisit, AssocVisitIndex , AssocForm, AssocFormIndex,
"signed") then
 ' associated to signed form
 endif
Next
```

## GetCurPath()

The GetCurPath method returns the RefName path of the context in which the rule is running.

## Example

```
val = Patient.GetValue(Patient.GetCurPath(), "", 0,0,0)
```

## GetCurrentFormIndex()

The GetCurrentFormIndex method returns 0 or a number containing the index of the current instance of a repeating form. Use this method to determine which instance of a repeating form is being changed, or whether a new instance is being added on Submit.

During the first submit of a repeating form instance, GetCurrentFormIndex returns 0. Thereafter it returns the index of the instance.

### Example

In this example, GetCurrentFormIndex passes the form index of the current instance of the repeating form to the *GetAssociationCount* (see "*GetAssociationCount(RefNamePath, VisitIndex, FormIndex)*" on page 258) method.

```

patient.RulePassed = 1
Patient.AddNamedValue "QUERYTEXT", "At least one of the Associated forms is missing data"
nAssociations = Patient.GetAssociationCount(Patient.GetCurPath(), Patient.GetCurrentVisitIndex(),
Patient.GetCurrentFormIndex())

for count = 1 to nAssociations
 Assoc = Patient.GetAssociationValue(count)
 AssocVisit = Assoc(0)
 AssocForm = Assoc(1)
 AssocVisitIndex = Assoc(2)
 AssocFormIndex = Assoc(3)
 'Do something with the association. Example: see if data is missing
 if(Patient.FormHasStateRF(AssocVisit, AssocVisitIndex , AssocForm, AssocFormIndex, "missing"))
 then
 patient.RulePassed = 0
 end if
Next

```

## GetCurrentISRowNum()

The GetCurrentISRowNum method returns a number containing the index of the current itemset row. If the current row is a new row, the method returns 0. Use this method to determine which row of an itemset is being changed, or whether a new row is being added on Submit.

### Example

This rule script, attached to the Adverse Experience form, captures information about the Adverse Experience being entered into the form. The variables are then passed to the execution plan using the AddNamedValue method.

```

ISrownum = Cint(patient.GetCurrentISRowNum())
AdverseExp =
Cstr(patient.GetValue("0.CommonCRF.AE.AE.AE.AEDESC.AEDESCTEXT", "", 0, 0, 0))
sitemnemonic = Cstr(patient.GetSiteMnemonic())
screeningnumber = Cstr(patient.GetScreeningNumber())
patientnumber = Cstr(patient.GetPatientNumber())
servertime = Cdate(patient.GetServerTime())

```

```
sitetime = Cdate(patient.GetSiteTime())
```

```
patient.AddNamedValue "passISrownum", ISrownum
patient.AddNamedValue "passAE", AdverseExp
patient.AddNamedValue "passitemnemonic", itemnemonic
patient.AddNamedValue "passcreeningnumber", screeningnumber
patient.AddNamedValue "passpatientnumber", patientnumber
patient.AddNamedValue "passservertime", servertime
patient.AddNamedValue "passsitetime", sitetime
```

An event for the above rule sets off an execution plan, which checks the itemset row number to make sure that the Adverse Experience being reported is a new one (value = 0). If the Adverse Experience is new, an email is sent out with several different pieces of information.

```
getISrownum = Cint(global.GetNamedValue("passISrownum"))
```

```
If (getISrownum = 0) then
 getAE = cstr(global.GetNamedValue("passAE"))
 getsitemnemonic = cstr(global.GetNamedValue("passitemnemonic"))
 getscreeningnumber = cstr(global.GetNamedValue("passcreeningnumber"))
 getpatientnumber = cstr(global.GetNamedValue("passpatientnumber"))
 getservertime = cdate(global.GetNamedValue("passservertime"))
 getsitetime = cdate(global.GetNamedValue("passsitetime"))
 emailsub = "Adverse Experience Notification: " & getAE
 emailtxt = "An Adverse Experience has occurred.
 Adverse Experience = " &getAE&,
 Site = " &getsitemnemonic& ",
 Site Time = " &getsitetime& ",
 Server Time = " &getservertime& ",
 Screening Number = " &getscreeningnumber& ",
 Patient Number = " &getpatientnumber
 global.EMailToInternetUser "user@pf.com", emailsub , emailtxt
end if
```

## GetCurrentVisitIndex()

The GetCurrentVisitIndex method returns a number containing the index of the current visit. Use this method to determine which visit of an unscheduled visit series is being changed, or whether a new visit is being added on Submit.

## Example

This script checks whether the visit date on the current unscheduled visit is later than the previous visit date and generates a query if the visit current visit date is earlier.

```
VisitIndex = Cint(patient.GetCurrentVisitIndex())

valnewvisit = Cdate(Patient.GetValue(Patient.GetCurPath() & "." &
 Patient.GetArgument("Controlname"), "", 0, 0, VisitIndex))
valoldvisit = Cdate(Patient.GetValue(Patient.GetCurPath() & "." &
 Patient.GetArgument("Controlname"), "", 0, 0, VisitIndex- 1))

difference = DateDiff("n", valnewvisit, valoldvisit)

if (VisitIndex = 1) then
 result.RulePassed=1
else
 if (difference > 0) then
 result.RulePassed=0
 patient.AddNamedValue "QUERYTEXT", "This date is earlier than the date
 of the previous visit."
 patient.AddNamedValue "QUERYSTATE", "Open"
 Else
 result.RulePassed=1
 end if
end if
```

## GetDefaultValue(RefNamePath)

The GetDefaultValue method is the same as the *GetValue* (see "*GetValue(REFNAMEPath,AttributeName,Index,ItemsetIndex,VisitIndex)*" on page 279) method, with default parameters. GetDefaultValue assumes the following default values for AttributeName, Index, ItemsetIndex, and VisitIndex:

- AttributeName="VALUE"—Method returns the value of the form component
- Index="0"—Form component is not a multiple- selection component
- ItemsetIndex="0"—Current itemset row, or item is not in an itemset
- VisitIndex="0"—Current visit

You only need to specify the RefNamePath of the item and the method returns the normalized value of the data entered in the specified form component. If the item is not normalized, the method returns the entered value.

## Arguments

### RefNamePath

String that identifies the form component to get. Each REFNAME in the string is the REFNAME specified in the .xml file that contains the definition of the form component. The REFNAMEPath must be fully qualified; only the patient component of the path defaults to the current patient. The REFNAMEPath argument is in the following format:

(0.Visit.Form.Section.Itemset.Item[.control[.control...]])

- *0*—*Current patient.*
- *Visit*—REFNAME of the visit, as specified in the .xml file that contains the visit definition.
- *Form*—REFNAME of the form, as specified in the .xml file that contains the form definition.
- *Section*—REFNAME of the section, as specified in the .xml file that contains the section definition.
- *Itemset*—REFNAME of the itemset, as specified in the .xml file that contains the itemset definition. An itemset is a group of data items that repeat. For example, a concomitant medication CRF might contain an itemset to capture the same set of data items about each concomitant medication the patient is using.
- *Item*—REFNAME of the item, as specified in the .xml file that contains the item definition.
- *Control*—REFNAME of the control, as specified in the .xml file that contains the control definition. To access an component of a group control, refer to each parent control in which the child component is nested. For example, to address one of two text controls within a group control, enter the REFNAME of the group control followed by the REFNAME of the text control, and separate the names with periods, as follows: *GroupControlRefname.TextControlRefname.*

### Example

```
Value = Patient.GetDefaultValue(Patient.GetCurPath())
```

```
If (Value>MinVal and Value<MaxVal) Then
 Result.RulePassed = 1
Else
 Result.RulePassed = 0
End If
```

## GetEnteredValue(REFNAMEPath,AttributeName,Index,ItemSetIndex,VisitIndex)

The GetEnteredValue method returns the value of the data entered in the specified form component. GetEnteredValue is very similar to the *GetValue* (see "*GetValue(REFNAMEPath,AttributeName,Index,ItemsetIndex,VisitIndex)*" on page 279) method, which returns the normalized value of the data entered in the specified form component. The difference between these methods applies to form components that are defined with units.

The MedML Installer utility enables unit form components to be defined with reference to other units, so that data can be displayed online as one unit and stored in the database, after conversion, as the other. For example, you can design a form so that a CRA can choose whether to enter a weight in ounces or pounds, and specify that the base unit is ounces. The InForm application stores the value as entered, whether it is ounces or pounds. If the entered value is pounds, the InForm application converts it to ounces, using a conversion rule that is attached to the unit definition, and stores the value in ounces in the database. The unit in which data is stored in the database after conversion is called the base unit, and the data value, after conversion to the base unit, is called the normalized value.

When you want to check the entered value of a data item, use GetEnteredValue. When you must get the normalized value of a data item, use GetValue.

**Note:** When you want to check a data value on a **repeating form**, or to attach a rule to a repeating form in order to reference a value on a non-repeating form, use *GetValueRF* (see "*GetValueRF(REFNAMEPath,AttributeName,Index,ItemsetIndex,VisitIndex,FormIndex)*" on page 281) or *GetEnteredValueRF* (see "*GetEnteredValueRF(REFNAMEPath,AttributeName,Index,ItemsetIndex,VisitIndex,FormIndex)*" on page 269)

## Arguments

REFNAMEPath String that identifies the form component to get. Each REFNAME in the string is the REFNAME specified in the .xml file that contains the definition of the form component. The REFNAMEPath must be fully qualified; only the patient component of the path defaults to the current patient. The REFNAMEPath argument is in the following format: (0.Visit.Form.Section.Itemset.Item[.control[.control...]])

- *0*—Current patient.
- *Visit*—REFNAME of the visit, as specified in the .xml file that contains the visit definition.
- *Form*—REFNAME of the form, as specified in the .xml file that contains the form definition.
- *Section*—REFNAME of the section, as specified in the .xml file that contains the section definition.
- *Itemset*—REFNAME of the itemset, as specified in the .xml file that contains the itemset definition. An itemset is a group of data items that repeat. For example, a concomitant medication CRF might contain an itemset to capture the same set of data items about each concomitant medication the patient is using.
- *Item*—REFNAME of the item, as specified in the .xml file that contains the item definition.



- *Control*—REFNAME of the control, as specified in the .xml file that contains the control definition. To access an component of a group control, refer to each parent control in which the child component is nested. For example, to address one of two text controls within a group control, enter the REFNAME of the group control followed by the REFNAME of the text control, and separate the names with periods, as follows: *GroupControlRefname.TextControlRefname*.

**AttributeName** String that specifies the type of information to return, specified in double quotes (""). Valid AttributeNames are:

- **VALUE**—Gets the value of the form component. This attribute is the default; you can indicate it with empty double quotes.
- **DATESTRING**—Returns only the date portion of a *DateTimeControl* (on page 105).
- **TIMESTRING**—Returns only the time portion of a *DateTimeControl*.
- **DATETIMESTRING**—Returns both the date and the time portions of a *DateTimeControl*.
- **SELECTED**—Used with the **Index** attribute to indicate whether the specified index is selected.
- **COUNT**—Specifies the number of items in the control that are selected; for example, if the control is a multiple-selection list box, **COUNT** returns the number of selected list items.
- **UNIT**—Returns the unit symbol text, if selected.
- **VALIDDATEMAP** — Number that specifies whether a date is complete. The script uses the **VALIDDATEMAP** attribute of the **GetValue** method on the **Patient** object. **VALIDDATEMAP** returns an integer that contains a bitmask indicating which elements of the *DateTimeControl* are complete:
  - **Year**—1
  - **Month**—2
  - **Day**—4
  - **Hour**—8
  - **Minute**—16
  - **Second**—32
- **UNKNOWNDATEMAP** — Number that specifies whether a date has components that are marked Unknown. The script uses the **UNKNOWNDATEMAP** attribute of the **GetValue** method on the **Patient** object. **UNKNOWNDATEMAP** returns an integer that contains a bitmask indicating which elements of the *DateTimeControl* are unknown:
  - **Year**—1
  - **Month**—2
  - **Day**—4
  - **Hour**—8
  - **Minute**—16
  - **Second**—32

**Note:** If you set a bit in the **UNKNOWNDATEMAP** value for a date component, the corresponding bit is not set in the **VALIDDATEMAP** value.

**Index**

Number, starting with 0, that specifies an index into an array of the selected items in a multiple-value control such as a list of check boxes. To use the Index attribute, either first return the COUNT of the control to determine how many items are selected, or find the selected items by looping through the list and creating an array of the selected items.

**ItemSetIndex**

Number that specifies an index into the items that make up an itemset. An itemset is a group of data items that repeat. For example, a concomitant medication CRF might contain an itemset to capture the same set of data items about each concomitant medication the patient is using. Valid ItemSetIndex values are 0 or greater:

- 0—Indicates that the control is not an itemset or, if the control is an itemset, references the current data item within the itemset. For example, if the user presses Submit after entering data in the third data item in the itemset, 0 sets the index to the third data item.
- Any number greater than 0—Explicitly indexes the specified item within the itemset.

**VisitIndex** Number that specifies an index into a repeating formset. A repeating formset is most often used for unscheduled visits in which the same data is recorded each time the visit occurs. Valid VisitIndex values are 0 or greater:

- 0—Indicates that the formset is not repeating or, if the formset is repeating, references the current formset.
- Any number greater than 0—Explicitly indexes the specified occurrence of the formset.

**Examples****Example 1: Checking the value of a text component**

The following example checks the value of the TEMPITEM component in the VS section of the VITALS form. If the value submitted in this component is within the specified minimum and maximum values, the rule is satisfied; otherwise the rule is not satisfied.

```
Control = "0.Visit2.VITALS.VS.0.TEMPITEM"
MinVal = 95
MaxVal = 105
```

```
'-----
Value = Patient.GetEnteredValue(Control, "", 0,0,0)
```

```
If (Value>MinVal and Value<MaxVal) Then
 Result.RulePassed = 1
Else
 Result.RulePassed = 0
End If
```

**Example 2: Addressing data in a compound control**

This example shows how to address data in a compound control made up of text boxes within a group box. The script checks the value of the SYSTEXT and DIASTEXT components, which are nested in the BREADINGGROUP group of the BREADINGITEM item.

```
Control1 = "0.Visit2.VITALS.VS.0.BPREADINGITEM.BPREADINGGROUP.SYSTEXT"
```

```
Control2 = "0.Visit2.VITALS.VS.0.BPREADINGITEM.BPREADINGGROUP.DIASTEXT"
```

```
'-----
Value1 = Patient.GetEnteredValue(Control1, "", 0,0,0)
Value2 = Patient.GetEnteredValue(Control2, "", 0,0,0)
```

### Example 3: Determining which radio button is selected

This example shows how to determine which radio button is selected in a compound control.

```
X = Patient.GetEnteredValue("0.Visit1.DEM.DEM.0.RACE", "SELECTED",
"RACEPULLDOWN",0,0)
Y = Patient.GetEnteredValue("0.Visit1.DEM.DEM.0.RACE", "SELECTED","RACETEXT",0,0)
```

```
If X = 1
 Then val = Patient.GetEnteredValue("0.Visit1.DEM.DEM.0.RACE", "",0,0,0)
 If val = (test against integer)
 End If
```

```
Else if Y = 1
 Then val = Patient.GetEnteredValue("0.Visit1.DEM.DEM.0.RACE", "",0,0,0)
 If val = (test against string)
 End If
```

```
End If
```

### Example 4: Getting a specific selected value in a multiple- selection list

This example returns the number of selected items in the WHATSMOKED multiple-selection list and then tests each selected item to determine if the Cigars item is selected.

```
nCount=Patient.GetValue("0.Visit1.DEM.SH.0.WHATSMOKED", "COUNT", 0,0,0)
i=1
bCigars=false
```

```
Do Until i < nCount + 1
 val=Patient.GetValue("0.Visit1.DEM.SH.0.WHATSMOKED", "", i,0,0)
 if(val="cigars") then
 bCigars=true
 Exit Do
 end if
Loop
```

```
if(bCigars) then
 Result.RulePassed=1
else
 Result.RulePassed=0
End if
```

### Example 5: Referring to a data item in an itemset

The following example shows how to refer to a data item within an itemset. The script checks the

value of the MEASUREITEM component, which is a data item in the AE itemset.

```
Control = "0.Visit2.AE.AE.AE.MEASUREITEM"
```

```
MinVal = 40
```

```
'-----
```

'This statement gets the value of MEASUREITEM in the current row,  
'the row the user just submitted or changed.

```
Value = Patient.GetEnteredValue(Control, "", 0,0,0)
```

'This statement gets the value of MEASUREITEM in the third row  
'of the itemset, by using the ItemsetIndex.

```
'Value = Patient.GetEnteredValue(Control, "", 0,3,0)
```

```
If (Value>MinVal) Then
```

```
Result.RulePassed = 1
```

```
Else
```

```
Result.RulePassed = 0
```

```
End If
```

### Example 6: Accessing unscheduled visits

The following examples show how to refer to data in a specific occurrence of a form that occurs in each unscheduled visit. These examples obtain data from the PULSERATETEXT data item in the second unscheduled visit. In the first example, the unscheduled visit is the current visit; in the second, the unscheduled visit is not the current visit.

```
Control = "0.UnschVisit.VS.VS.0.PULSERATE.PULSERATETEXT"
```

```
Value = Patient.GetEnteredValue(Control, "", 0,0,0)
```

```
Control = "0.UnschVisit.VS.VS.0.PULSERATE.PULSERATETEXT"
```

```
Value = Patient.GetEnteredValue(Control, "", 0,0,2)
```

### Example 7: Verifying required dates are complete

The following example uses VALIDDATEMAP to verify that the required parts of a date are completed in the form. A value is assigned to each component that makes up a complete date. In the example below, a value of 4 for day, 2 for month and 1 for year adds up to a total of 7. If data is present in each of these components, then PFCOMPLETEDATE = 7, and the rule passes. If any of these components are missing data, then PFCOMPLETEDATE will not equal 7 and the rule fails.

```
strItem = "0.Visit1.DEM.DEM.0.DEMDOB"
```

```
PFYEAR=1
```

```
PFMONTH=2
```

```
PFDAY=4
```

```
PFCOMPLETEDATE = PFYEAR Or PFMONTH Or PFDAY 'PFCOMPLETEDATE=7
```

```
testDateMap = Patient.GetValue(strItem, "VALIDDATEMAP", 0, 0, 0)
```

```
If testDateMap=PFCOMPLETEDATE Then
```

```
Result.RulePassed = 1
```

```
Else
```

```
Result.RulePassed = 0 End If
```

### Example 8: Looping through ItemsetIndex

The following example uses `GetItemsetCount` in `GetValue` to loop through `ItemsetIndex`.

```
ISidx = 1
vPgPath = Patient.GetArgument("PgCtrlPath")
vIScnt = CInt(Patient.GetItemsetCount(vPgPath,0))
Patient.RulePassed = 0 'failed unless we find a match
Do While (ISidx <= vIScnt)
 [CODE]
 ISidx = ISidx + 1
Loop
```

### GetEnteredValueRF(REFNAMEPath,AttributeName,Index,ItemsetIndex,VisitIndex, FormIndex)

In instances of a repeating form, the `GetEnteredValueRF` method returns the value of the data entered in the specified form component. The `GetEnteredValueRF` method is identical to the *GetEnteredValue* (see "*GetEnteredValue(REFNAMEPath,AttributeName,Index,ItemSetIndex,VisitIndex)*" on page 264) method, except that it is intended for use in repeating forms. Therefore, it has an additional argument, `FormIndex`, which enables you to reference a specific instance of the repeating form.

**Note:** Use `GetEnteredValueRF` either when you check a value on a repeating form or when you attach a rule to a repeating form and reference a value in a non-repeating form.

### Arguments

`REFNAMEPath` String that identifies the form component to get. Each `REFNAME` in the string is the `REFNAME` specified in the `.xml` file that contains the definition of the form component. The `REFNAMEPath` must be fully qualified; only the patient component of the path defaults to the current patient. The `REFNAMEPath` argument is in the following format:  
(0.Visit.Form.Section.Itemset.Item[.control[.control...]])

- *0*—Current patient.
- *Visit*—`REFNAME` of the visit, as specified in the `.xml` file that contains the visit definition.
- *Form*—`REFNAME` of the form, as specified in the `.xml` file that contains the form definition.
- *Section*—`REFNAME` of the section, as specified in the `.xml` file that contains the section definition.
- *Itemset*—`REFNAME` of the itemset, as specified in the `.xml` file that contains the itemset definition. An itemset is a group of data items that repeat. For example, a concomitant medication CRF might contain an itemset to capture the same set of data items about each concomitant medication the patient is using.
- *Item*—`REFNAME` of the item, as specified in the `.xml` file that contains the item definition.

- *Control*—REFNAME of the control, as specified in the .xml file that contains the control definition. To access an component of a group control, refer to each parent control in which the child component is nested. For example, to address one of two text controls within a group control, enter the REFNAME of the group control followed by the REFNAME of the text control, and separate the names with periods, as follows: *GroupControlRefname.TextControlRefname*.

**AttributeName** String that specifies the type of information to return, specified in double quotes (""). Valid AttributeNames are:

- **VALUE**—Gets the value of the form component. This attribute is the default; you can indicate it with empty double quotes.
- **DATESTRING**—Returns only the date portion of a *DateTimeControl* (on page 105).
- **TIMESTRING**—Returns only the time portion of a *DateTimeControl*.
- **DATETIMESTRING**—Returns both the date and the time portions of a *DateTimeControl*.
- **SELECTED**—Used with the **Index** attribute to indicate whether the specified index is selected.
- **COUNT**—Specifies the number of items in the control that are selected; for example, if the control is a multiple-selection list box, **COUNT** returns the number of selected list items.
- **UNIT**—Returns the unit symbol text, if selected.
- **VALIDDATEMAP** — Number that specifies whether a date is complete. The script uses the **VALIDDATEMAP** attribute of the **GetValue** method on the **Patient** object. **VALIDDATEMAP** returns an integer that contains a bitmask indicating which elements of the *DateTimeControl* are complete:
  - **Year**—1
  - **Month**—2
  - **Day**—4
  - **Hour**—8
  - **Minute**—16
  - **Second**—32
- **UNKNOWNDATEMAP** — Number that specifies whether a date has components that are marked Unknown. The script uses the **UNKNOWNDATEMAP** attribute of the **GetValue** method on the **Patient** object. **UNKNOWNDATEMAP** returns an integer that contains a bitmask indicating which elements of the *DateTimeControl* are unknown:
  - **Year**—1
  - **Month**—2
  - **Day**—4
  - **Hour**—8
  - **Minute**—16
  - **Second**—32

**Note:** If you set a bit in the **UNKNOWNDATEMAP** value for a date component, the corresponding bit is not set in the **VALIDDATEMAP** value.

## Index

Number, starting with 0, that specifies an index into an array of the selected items in a multiple-value control such as a list of check boxes. To use the Index attribute, either first return the COUNT of the control to determine how many items are selected, or find the selected items by looping through the list and creating an array of the selected items.

## ItemsetIndex

Number that specifies an index into the items that make up an itemset. An itemset is a group of data items that repeat. For example, a concomitant medication CRF might contain an itemset to capture the same set of data items about each concomitant medication the patient is using. Valid ItemsetIndex values are 0 or greater:

- 0—Indicates that the control is not an itemset or, if the control is an itemset, references the current data item within the itemset. For example, if the user presses Submit after entering data in the third data item in the itemset, 0 sets the index to the third data item.
- Any number greater than 0—Explicitly indexes the specified item within the itemset.

**VisitIndex** Number that specifies an index into a repeating formset. A repeating formset is most often used for unscheduled visits in which the same data is recorded each time the visit occurs. Valid VisitIndex values are 0 or greater:

- 0—Indicates that the formset is not repeating or, if the formset is repeating, references the current formset.
- Any number greater than 0—Explicitly indexes the specified occurrence of the formset.

**FormIndex** Number that specifies an index into a repeating form.

- 0—References the current repeating form.
- 1—References a non-repeating form.
- Any number greater than 1—References the specified occurrence of the repeating form.

## Example

This example obtains the value of two controls on a nonrepeating form so it can use them in the current form, which is repeating.

```
Control1 = "0.Visit2.VITALS.VS.0.BPITEM.BPGROUP.SYSTEXT"
Control2 = "0.Visit2.VITALS.VS.0.BPITEM.BPGROUP.DIASTEXT"
Value1 = Patient.GetEnteredValueRF(Control1, "", 0,0,0,0)
Value2 = Patient.GetEnteredValueRF(Control2, "", 0,0,0,0)
```

## GetFormCount(Visit,VisitIndex,Form)

The GetFormCount method returns the number of repeating form instances that have been created for a specified visit.

### Arguments

**Visit** RefName of the formset in which the repeating form occurs.

#### VisitIndex

Number that specifies an index into a repeating formset. A repeating formset is most often used for unscheduled visits in which the same data is recorded each time the visit occurs. Valid VisitIndex values are 0 or greater:

- 0—Indicates that the formset is not repeating or, if the formset is repeating, references the current formset.
- Any number greater than 0—Explicitly indexes the specified occurrence of the formset.

**Form** RefName of the repeating form.

### Example

The following example tests whether the adverse experience being entered already exists. It uses the GetFormCount method to set the number of times to perform the test, based on the total number of AE instances.

```

patient.RulePassed = 1
Patient.AddNamedValue "QUERYTEXT", "This AE already exists"
path="0.CommonCRF.AE.sctAE.0.itmAEDiag.txtAEDiag"
isValid=patient.IsValidItemPathRF (path,0,0)
strVal=patient.GetValueRF(path,"",0,0,0,0)
intcurFormCount=patient.GetCurrentFormIndex()
intFormCount=patient.GetFormCount("CommonCRF",1,"AE")

for count = 1 to intFormCount
intIsDel=Patient.FormHasStateRF("CommonCRF",1,"AE",count,"deleted")
if (cint(count) <> cint(intcurFormCount)) and intIsDel=0 then
strVal2=patient.GetValueRF("0.CommonCRF.AE.sctAE.0.itmAEDiag.txtAEDiag","", 0, 0, 0,
count)
If Ucase(strVal) = Ucase(strVal2) then
patient.RulePassed = 0
exit for
End If
End If
Next

```



## GetID()

The GetID method returns the Id of the current patient. The Id is an internally assigned value by which the patient is known in the database.

## Example

The following example illustrates the use of the GetID method to load the properties of the Randomization object in preparation for generating a randomization kit number.

```
Randomization.SiteID = Patient.GetSiteID
Randomization.Type = 1
Randomization.Source = "SimpleCentral"
Randomization.PatientID = Patient.GetID
Randomization.Revision = 0
SeqNum = Randomization.GetNextKit(KitInfo)
Result.Result= SeqNum + " / " + KitInfo
```

## GetItemsetCount(REFNAMEPath,VisitIndex)

The GetItemsetCount method returns the number of rows currently entered in an itemset.

## Arguments

REFNAMEPath String that identifies the form component to get. Each REFNAME in the string is the REFNAME specified in the .xml file that contains the definition of the form component. The REFNAMEPath must be fully qualified; only the patient component of the path defaults to the current patient. The REFNAMEPath argument is in the following format:

(0.Visit.Form.Section.Itemset.Item[control[control...]])

- *0*—Current patient.
- *Visit*—REFNAME of the visit, as specified in the .xml file that contains the visit definition.
- *Form*—REFNAME of the form, as specified in the .xml file that contains the form definition.
- *Section*—REFNAME of the section, as specified in the .xml file that contains the section definition.
- *Itemset*—REFNAME of the itemset, as specified in the .xml file that contains the itemset definition. An itemset is a group of data items that repeat. For example, a concomitant medication CRF might contain an itemset to capture the same set of data items about each concomitant medication the patient is using.
- *Item*—REFNAME of the item, as specified in the .xml file that contains the item definition.
- *Control*—REFNAME of the control, as specified in the .xml file that contains the control definition. To access an component of a group control, refer to each parent control in which the child component is nested. For example, to address one of two text controls within a group control, enter the REFNAME of the group control followed by the REFNAME of the text control, and separate the names with periods, as follows: *GroupControlRefname.TextControlRefname*.

**VisitIndex** Number that specifies an index into a repeating formset. A repeating formset is most often used for unscheduled visits in which the same data is recorded each time the visit occurs. Valid VisitIndex values are 0 or greater:

- *0*—Indicates that the formset is not repeating or, if the formset is repeating, references the

current formset.

- Any number greater than 0—Explicitly indexes the specified occurrence of the formset.

## Example

The following example uses AETEXTITEM, the first data item in the AE itemset, to determine the number of rows entered in the AE itemset. Note that you must cast the result of GetItemsetCount as an integer.

```
CheckItem = "0.Visit2.AE.AE.AE.AETEXTITEM"
```

```
CheckCount = Cint(Patient.GetItemsetCount(CheckItem,0))
```

## GetItemsetCountRF(REFNAMEPath,VisitIndex,FormIndex)

The GetItemsetCountRF method returns the number of rows currently entered in an itemset that occurs in the specified instance of a repeating form.

### Arguments

**REFNAMEPath** String that identifies the form component to get. Each REFNAME in the string is the REFNAME specified in the .xml file that contains the definition of the form component. The REFNAMEPath must be fully qualified; only the patient component of the path defaults to the current patient. The REFNAMEPath argument is in the following format:

(0.Visit.Form.Section.Itemset.Item[control[control...]])

- *0*—Current patient.
- *Visit*—REFNAME of the visit, as specified in the .xml file that contains the visit definition.
- *Form*—REFNAME of the form, as specified in the .xml file that contains the form definition.
- *Section*—REFNAME of the section, as specified in the .xml file that contains the section definition.
- *Itemset*—REFNAME of the itemset, as specified in the .xml file that contains the itemset definition. An itemset is a group of data items that repeat. For example, a concomitant medication CRF might contain an itemset to capture the same set of data items about each concomitant medication the patient is using.
- *Item*—REFNAME of the item, as specified in the .xml file that contains the item definition.
- *Control*—REFNAME of the control, as specified in the .xml file that contains the control definition. To access an component of a group control, refer to each parent control in which the child component is nested. For example, to address one of two text controls within a group control, enter the REFNAME of the group control followed by the REFNAME of the text control, and separate the names with periods, as follows: *GroupControlRefname.TextControlRefname*.

**VisitIndex** Number that specifies an index into a repeating formset. A repeating formset is most often used for unscheduled visits in which the same data is recorded each time the visit occurs. Valid VisitIndex values are 1 or greater:

- 1—Indicates that the formset is not repeating or, if the formset is repeating, references the current formset.
- Any number greater than 1—Explicitly indexes the specified occurrence of the formset.

**FormIndex** Number that specifies an index into a repeating form.

- 0—References the current repeating form.
- 1—References a non-repeating form.
- Any number greater than 1—References the specified occurrence of the repeating form.

## Example

This example counts the number of itemset rows in the nonrepeating AE form so it can use them in another form, which is repeating.

```
CheckItem = "0.Visit2.AE.AE.AE.AETEXTITEM"
CheckCount = Cint(Patient.GetItemsetCountRF(CheckItem,1,1))
```

## GetItemsetRowID(REFNAMEPath,VisitIndex)

The GetItemsetRowID method returns the uniquely identifying row ID of the itemset row.

## Arguments

**REFNAMEPath** String that identifies the form component to get. Each REFNAME in the string is the REFNAME specified in the .xml file that contains the definition of the form component. The REFNAMEPath must be fully qualified; only the patient component of the path defaults to the current patient. The REFNAMEPath argument is in the following format:

```
(0.Visit.Form.Section.Itemset.Item[control[control...]])
```

- *0* — *Current patient.*
- *Visit* — REFNAME of the visit, as specified in the .xml file that contains the visit definition.
- *Form* — REFNAME of the form, as specified in the .xml file that contains the form definition.
- *Section* — REFNAME of the section, as specified in the .xml file that contains the section definition.
- *Itemset* — REFNAME of the itemset, as specified in the .xml file that contains the itemset definition. An itemset is a group of data items that repeat. For example, a concomitant medication CRF might contain an itemset to capture the same set of data items about each concomitant medication the patient is using.
- *Item* — REFNAME of the item, as specified in the .xml file that contains the item definition.
- *Control* — REFNAME of the control, as specified in the .xml file that contains the control definition. To access an component of a group control, refer to each parent control in which the child component is nested. For example, to address one of two text controls within a group control, enter the REFNAME of the group control followed by the REFNAME of the text control, and separate the names with periods, as follows: *GroupControlRefname.TextControlRefname.*

**VisitIndex** Number that specifies an index into a repeating formset. A repeating formset is most often used for unscheduled visits in which the same data is recorded each time the visit occurs. Valid VisitIndex values are 0 or greater:

- *0* — Indicates that the formset is not repeating or, if the formset is repeating, references the current formset.
- Any number greater than 0 — Explicitly indexes the specified occurrence of the formset.

## Example

The following example retrieves the itemset row ID of AETEXTITEM, the first data item in the AE itemset.

```
CheckItem = "0.Visit2.AE.AE.AE.AETEXTITEM"
CheckID = Cint(Patient.GetItemsetRowID(CheckItem,0))
```

## GetPatientNumber()

The GetPatientNumber method returns the number assigned to the current patient. The patient number is a value assigned to a patient during enrollment.

## Example

See the example for *GetCurrentISRowNum()* (on page 260)

## GetRefName(ObjType)

The GetRefName method returns a string containing the RefName of the specified object type in the context in which the rule is run.

## Arguments

**ObjType** One of the following: ("Visit","Form","Section","Item","Itemset")

## GetRefPath(ObjType)

The GetRefPath method returns a string containing the RefName path of the specified object type in the context in which the rule is run. For example, if you specify "Section," GetRefPath returns a RefName path of 0.*VisitRefName.FormRefName.SectionRefName*.

## Arguments

**ObjType** One of the following: ("Visit","Form","Section","Item","Itemset")

## GetScreeningNumber()

The GetScreeningNumber method returns a string containing the current patient's screening number.

## Example

See the example for *GetCurrentISRowNum()* (on page 260)

## GetServerTime()

The GetServerTime method returns a string containing the datetime value of the current server as GMT.

### Example

See the example for *GetCurrentISRowNum()* (on page 260)

## GetSiteID()

The GetSiteID method returns the Id of the current patient's site. The site Id is an internally assigned value by which the site is known in the database.

### Example

```
SiteID = Patient.GetSiteID()
If SiteID...
```

## GetSiteMnemonic()

The GetSiteMnemonic method returns a string containing the site mnemonic for the current patient's site. This method, which is similar to the *GetSiteName* (see "*GetSiteName()*" on page 278) method, provides an abbreviated way to identify a site.

### Example

See the example for *GetCurrentISRowNum()* (on page 260)

## GetSiteName()

The GetSiteName method returns the name of the current patient's site.

### Example

```
SiteName = Patient.GetSiteName()
If SiteName...
```

## GetSiteTime()

The GetSiteTime method returns the datetime value of the server in the current site's time zone. This method is useful for insuring that a future date or time is not entered on a form.

### Example

See the example for *GetCurrentISRowNum()* (on page 260)

## GetValue(REFNAMEPath,AttributeName,Index,ItemsetIndex,VisitIndex)

The GetValue method returns the normalized value of the data entered in the specified form component. If the item is not normalized, the method returns the entered value. GetValue is very similar to the *GetEnteredValue* (see "*GetEnteredValue(REFNAMEPath,AttributeName,Index,ItemSetIndex,VisitIndex)*" on page 264) method, which returns the entered value of the data entered in the specified form component.

The difference between these methods applies to form components that are defined as units.

The MedML Installer utility enables unit form components to be defined with reference to other units, so that data can be displayed online as one unit and stored in the database, after conversion, as the other. For example, you can design a form so that a CRA can choose whether to enter a weight in ounces or pounds, and specify that the base unit is ounces. The InForm application stores the value as entered, whether it is ounces or pounds. If the entered value is pounds, the InForm application converts it to ounces, using a conversion rule that you attach to the unit definition, and stores the value in ounces in the database. The unit in which data is stored in the database after conversion is called the base unit, and the data value, after conversion to the base unit, is called the normalized value.

When you want to check a data value as **entered**, use GetEnteredValue. When you must get the normalized value of a data item, use GetValue.

**Note:** When you want to check a data value on a **repeating form**, or to attach a rule to a repeating form in order to reference a value on a non-repeating form, use *GetValueRF* (see "*GetValueRF(REFNAMEPath,AttributeName,Index,ItemsetIndex,VisitIndex,FormIndex)*" on page 281) or *GetEnteredValueRF* (see "*GetEnteredValueRF(REFNAMEPath,AttributeName,Index,ItemsetIndex,VisitIndex,FormIndex)*" on page 269)

## Arguments

**REFNAMEPath** String that identifies the form component to get. Each REFNAME in the string is the REFNAME specified in the .xml file that contains the definition of the form component. The REFNAMEPath must be fully qualified; only the patient component of the path defaults to the current patient. The REFNAMEPath argument is in the following format:  
(0.Visit.Form.Section.Itemset.Item[.control[.control...]])

- *0*—*Current patient.*
- *Visit*—REFNAME of the visit, as specified in the .xml file that contains the visit definition.
- *Form*—REFNAME of the form, as specified in the .xml file that contains the form definition.
- *Section*—REFNAME of the section, as specified in the .xml file that contains the section definition.
- *Itemset*—REFNAME of the itemset, as specified in the .xml file that contains the itemset definition. An itemset is a group of data items that repeat. For example, a concomitant medication CRF might contain an itemset to capture the same set of data items about each concomitant medication the patient is using.
- *Item*—REFNAME of the item, as specified in the .xml file that contains the item definition.

- *Control*—REFNAME of the control, as specified in the .xml file that contains the control definition. To access an component of a group control, refer to each parent control in which the child component is nested. For example, to address one of two text controls within a group control, enter the REFNAME of the group control followed by the REFNAME of the text control and separated by periods, as follows: *GroupControlRefname.TextControlRefname*.

**AttributeName** String that specifies the type of information to return, specified in double quotes (""). Valid AttributeNames are:

- **VALUE**—Gets the value of the form component. This attribute is the default; you can indicate it with empty double quotes.
- **DATESTRING**—Returns only the date portion of a *DateTimeControl* (on page 105).
- **TIMESTRING**—Returns only the time portion of a *DateTimeControl*.
- **DATETIMESTRING**—Returns both the date and the time portions of a *DateTimeControl*.
- **SELECTED**—Used with the **Index** attribute to indicate whether the specified index is selected.
- **COUNT**—Specifies the number of items in the control that are selected; for example, if the control is a multiple-selection list box, **COUNT** returns the number of selected list items.
- **UNIT**—Returns the unit symbol text, if selected.
- **VALIDDATEMAP** — Number that specifies whether a date is complete. The script uses the **VALIDDATEMAP** attribute of the **GetValue** method on the **Patient** object. **VALIDDATEMAP** returns an integer that contains a bitmask indicating which elements of the *DateTimeControl* are complete:
  - **Year**—1
  - **Month**—2
  - **Day**—4
  - **Hour**—8
  - **Minute**—16
  - **Second**—32
- **UNKNOWNDATEMAP** — Number that specifies whether a date has components that are marked **Unknown**. The script uses the **UNKNOWNDATEMAP** attribute of the **GetValue** method on the **Patient** object. **UNKNOWNDATEMAP** returns an integer that contains a bitmask indicating which elements of the *DateTimeControl* are unknown:
  - **Year**—1
  - **Month**—2
  - **Day**—4
  - **Hour**—8
  - **Minute**—16
  - **Second**—32

**Note:** If you set a bit in the **UNKNOWNDATEMAP** value for a date component, the corresponding bit is not set in the **VALIDDATEMAP** value.



## Index

Number, starting with 0, that specifies an index into an array of the selected items in a multiple-value control such as a list of check boxes. To use the Index attribute, either first return the COUNT of the control to determine how many items are selected, or find the selected items by looping through the list and creating an array of the selected items.

## ItemSetIndex

Number that specifies an index into the items that make up an itemset. An itemset is a group of data items that repeat. For example, a concomitant medication CRF might contain an itemset to capture the same set of data items about each concomitant medication the patient is using. Valid ItemSetIndex values are 0 or greater:

- 0—Indicates that the control is not an itemset or, if the control is an itemset, references the current data item within the itemset. For example, if the user presses Submit after entering data in the third data item in the itemset, 0 sets the index to the third data item.
- Any number greater than 0—Explicitly indexes the specified item within the itemset.

**VisitIndex** Number that specifies an index into a repeating formset. A repeating formset is most often used for unscheduled visits in which the same data is recorded each time the visit occurs. Valid VisitIndex values are 0 or greater:

- 0—Indicates that the formset is not repeating or, if the formset is repeating, references the current formset.
- Any number greater than 0—Explicitly indexes the specified occurrence of the formset.

## Examples

See the examples listed for the *GetEnteredValue method* (see "*GetEnteredValue(REFNAMEPath,AttributeName,Index,ItemSetIndex,VisitIndex)*" on page 264).

## GetValueRF(REFNAMEPath,AttributeName,Index,ItemsetIndex,VisitIndex,FormIndex)

In instances of a repeating form, the GetValueRF method returns the normalized value of the data entered in the specified form component. If the item is not normalized, the method returns the entered value. The GetValueRF method is identical to the *GetValue* (see "*GetValue(REFNAMEPath,AttributeName,Index,ItemsetIndex,VisitIndex)*" on page 279) method, except that it is intended for use in repeating forms. Therefore, it has an additional argument, FormIndex, which enables you to reference a specific instance of the repeating form.

**Note:** Use GetValueRF either when you check a value on a repeating form or when you attach a rule to a repeating form and reference a value in a non-repeating form.

## Arguments

**REFNAMEPath** String that identifies the form component to get. Each REFNAME in the string is the REFNAME specified in the .xml file that contains the definition of the form component. The REFNAMEPath must be fully qualified; only the patient component of the path defaults to the current patient. The REFNAMEPath argument is in the following format:

(0.Visit.Form.Section.Itemset.Item[.control[.control...]])

- *0*—Current patient.
- *Visit*—REFNAME of the visit, as specified in the .xml file that contains the visit definition.
- *Form*—REFNAME of the form, as specified in the .xml file that contains the form definition.
- *Section*—REFNAME of the section, as specified in the .xml file that contains the section definition.
- *Itemset*—REFNAME of the itemset, as specified in the .xml file that contains the itemset definition. An itemset is a group of data items that repeat. For example, a concomitant medication CRF might contain an itemset to capture the same set of data items about each concomitant medication the patient is using.
- *Item*—REFNAME of the item, as specified in the .xml file that contains the item definition.
- *Control*—REFNAME of the control, as specified in the .xml file that contains the control definition. To access an component of a group control, refer to each parent control in which the child component is nested. For example, to address one of two text controls within a group control, enter the REFNAME of the group control followed by the REFNAME of the text control and separated by periods, as follows: *GroupControlRefname.TextControlRefname*.

**AttributeName** String that specifies the type of information to return, specified in double quotes (""). Valid AttributeNames are:

- **VALUE**—Gets the value of the form component. This attribute is the default; you can indicate it with empty double quotes.
- **DATESTRING**—Returns only the date portion of a *DateTimeControl* (on page 105).
- **TIMESTRING**—Returns only the time portion of a *DateTimeControl*.
- **DATETIMESTRING**—Returns both the date and the time portions of a *DateTimeControl*.
- **SELECTED**—Used with the **Index** attribute to indicate whether the specified index is selected.
- **COUNT**—Specifies the number of items in the control that are selected; for example, if the control is a multiple-selection list box, **COUNT** returns the number of selected list items.
- **UNIT**—Returns the unit symbol text, if selected.
- **VALIDDATEMAP** — Number that specifies whether a date is complete. The script uses the **VALIDDATEMAP** attribute of the **GetValue** method on the **Patient** object. **VALIDDATEMAP** returns an integer that contains a bitmask indicating which elements of the *DateTimeControl* are complete:
  - **Year**—1
  - **Month**—2
  - **Day**—4

- **Hour**—8
- **Minute**—16
- **Second**—32
- **UNKNOWNDATEMAP** — Number that specifies whether a date has components that are marked Unknown. The script uses the UNKNOWNDATEMAP attribute of the GetValue method on the Patient object. UNKNOWNDATEMAP returns an integer that contains a bitmask indicating which elements of the DateTimeControl are unknown:
  - **Year**—1
  - **Month**—2
  - **Day**—4
  - **Hour**—8
  - **Minute**—16
  - **Second**—32

**Note:** If you set a bit in the UNKNOWNDATEMAP value for a date component, the corresponding bit is not set in the VALIDDATEMAP value.

**Index**

Number, starting with 0, that specifies an index into an array of the selected items in a multiple-value control such as a list of check boxes. To use the Index attribute, either first return the COUNT of the control to determine how many items are selected, or find the selected items by looping through the list and creating an array of the selected items.

**ItemSetIndex**

Number that specifies an index into the items that make up an itemset. An itemset is a group of data items that repeat. For example, a concomitant medication CRF might contain an itemset to capture the same set of data items about each concomitant medication the patient is using. Valid ItemSetIndex values are 0 or greater:

- 0—Indicates that the control is not an itemset or, if the control is an itemset, references the current data item within the itemset. For example, if the user presses Submit after entering data in the third data item in the itemset, 0 sets the index to the third data item.
- Any number greater than 0—Explicitly indexes the specified item within the itemset.

**VisitIndex** Number that specifies an index into a repeating formset. A repeating formset is most often used for unscheduled visits in which the same data is recorded each time the visit occurs. Valid VisitIndex values are 0 or greater:

- 0—Indicates that the formset is not repeating or, if the formset is repeating, references the current formset.
- Any number greater than 0—Explicitly indexes the specified occurrence of the formset.

**FormIndex** Number that specifies an index into a repeating form.

- 0—References the current repeating form.
- 1—References a non-repeating form.
- Any number greater than 1—References the specified occurrence of the repeating form.

**Example**

The following example tests whether the text description of the adverse experience being entered in the txtAEDiag control already exists. It uses the GetValueRF method to obtain the value entered in the text control of the current form instance and in each succeeding form instance being tested so those values can be compared.

```

patient.RulePassed = 1
Patient.AddNamedValue "QUERYTEXT", "This AE already exists"
path="0.CommonCRF.AE.sctAE.0.itmAEDiag.txtAEDiag"
isValid=patient.IsValidItemPathRF (path,0,0)
strVal=patient.GetValueRF(path,"",0,0,0,0)
intcurFormCount=patient.GetCurrentFormIndex()
intFormCount=patient.GetFormCount("CommonCRF",1,"AE")

for count = 1 to intFormCount
intIsDel=Patient.FormHasStateRF("CommonCRF",1,"AE",count,"deleted")
if (cint(count) <> cint(intcurFormCount)) and intIsDel=0 then
strVal2=patient.GetValueRF("0.CommonCRF.AE.sctAE.0.itmAEDiag.txtAEDiag","", 0, 0, 0,
count)

```

```

If Ucase(strVal) = Ucase(strVal2) then
patient.RulePassed = 0
exit for
End If
End If
Next

```

## GetVisitCount(VisitRefName)

The GetVisitCount method returns the number of visits for the visit specified in VisitRefName. This method is used for unscheduled visits, and can be used to establish a constant for counting iterations through a loop of all repetitions of a visit.

## Arguments

**Visit RefName** The RefName of the visit for which you want to count number of visits.

## Example

```

visitname = Cstr(Patient.GetArgument("visitrefname"))
visitcount = Cint(patient.Getvisitcount(visitname))

```

## IsCBSSelected(RefNamePath,ItemSetIndex, VisitIndex)

The IsCBSSelected method checks to see if a child control in a CheckBoxControl is selected and returns a Boolean containing the state of the child control. True=checked, False=unchecked.

## Arguments

**RefNamePath** String that identifies the simple control of the checkbox item to be tested. Each REFNAME in the string is the REFNAME specified in the .xml file that contains the definition of the form component. The REFNAMEPath must be fully qualified; only the patient component of the path defaults to the current patient. The REFNAMEPath argument is in the following format: *(0.Visit.Form.Section.Itemset.Item[.control[.control...]])*

- *0*—Current patient.
- *Visit*—REFNAME of the visit, as specified in the .xml file that contains the visit definition.
- *Form*—REFNAME of the form, as specified in the .xml file that contains the form definition.
- *Section*—REFNAME of the section, as specified in the .xml file that contains the section definition.
- *Itemset*—REFNAME of the itemset, as specified in the .xml file that contains the itemset definition. An itemset is a group of data items that repeat. For example, a concomitant medication CRF might contain an itemset to capture the same set of data items about each concomitant medication the patient is using.
- *Item*—REFNAME of the item, as specified in the .xml file that contains the item definition.

- *Control*—REFNAME of the child control that contains the checkbox you want to test. To access a checkbox component of a group control, refer to each parent control in which the child component is nested. For example, to address one of two simple controls within a group control, enter the REFNAME of the group control followed by the REFNAME of the simple control, and separate the names with periods, as follows: *GroupControlRefname.SimpleControlRefname*.

**ItemSetIndex** Number that specifies an index into the items that make up an itemset. An itemset is a group of data items that repeat. For example, a concomitant medication CRF might contain an itemset to capture the same set of data items about each concomitant medication the patient is using. Valid ItemSetIndex values are 0 or greater:

- 0—Indicates that the control is not an itemset or, if the control is an itemset, references the current data item within the itemset. For example, if the user presses Submit after entering data in the third data item in the itemset, 0 sets the index to the third data item.
- Any number greater than 0—Explicitly indexes the specified item within the itemset.

**VisitIndex** Number that specifies an index into a repeating formset. A repeating formset is most often used for unscheduled visits in which the same data is recorded each time the visit occurs. Valid VisitIndex values are 0 or greater:

- 0—Indicates that the formset is not repeating or, if the formset is repeating, references the current formset.
- Any number greater than 0—Explicitly indexes the specified occurrence of the formset.

## Example

The following example checks each SimpleControl in a CheckBoxControl to determine whether it is selected. If all boxes are empty, it generates a query if the radio button for the CheckBoxControl was selected.

```

cbselected = "False"
valradio = Cstr(Patient.GetValue(Patient.GetCurPath() & ".SMOKEGROUPRADIO", "", 0, 0, 0))

If CBool(patient.IsCBSelected(Patient.GetCurPath() &
 ".SMOKEGROUPRADIO.SMOKEGROUP.SMOKECHECKBOX.CIGARETTE",0,0)) =
"True" then
 cbselected = "True"
End if

If CBool(patient.IsCBSelected(Patient.GetCurPath() &
 ".SMOKEGROUPRADIO.SMOKEGROUP.SMOKECHECKBOX.PIPE",0,0)) = "True" then
 cbselected = "True"
End if

If CBool(patient.IsCBSelected(Patient.GetCurPath() &
 ".SMOKEGROUPRADIO.SMOKEGROUP.SMOKECHECKBOX.CIGAR",0,0)) = "True"
then
 cbselected = "True"
End if

if valradio = "Smokes" then
 if cbselected = "False" then
 result.RulePassed=0

```

```

 patient.AddNamedValue "QUERYTEXT", "Please select at least one checkbox."
 patient.AddNamedValue "QUERYSTATE", "Open"
 Else
 result.RulePassed=1
 End if
Else
 result.RulePassed=1
End if

```

## IsCBSelectedRF(RefNamePath,ItemsetIndex,VisitIndex,FormIndex)

In an instance of a repeating form, the IsCBSelectedRF method checks to see if a child control in a CheckBoxControl is selected and returns a Boolean containing the state of the child control. True=checked, False=unchecked.

The IsCBSelectedRF method is identical to the IsCBSelected method, except that it is intended for use in repeating forms. Therefore, it has an additional argument, FormIndex, which enables you to reference a specific instance of the repeating form.

## Arguments

**RefNamePath** String that identifies the simple control of the checkbox item to be tested. Each REFNAME in the string is the REFNAME specified in the .xml file that contains the definition of the form component. The REFNAMEPath must be fully qualified; only the patient component of the path defaults to the current patient. The REFNAMEPath argument is in the following format: *(0.Visit.Form.Section.Itemset.Item[.control[.control...]])*

- *0*—Current patient.
- *Visit*—REFNAME of the visit, as specified in the .xml file that contains the visit definition.
- *Form*—REFNAME of the form, as specified in the .xml file that contains the form definition.
- *Section*—REFNAME of the section, as specified in the .xml file that contains the section definition.
- *Itemset*—REFNAME of the itemset, as specified in the .xml file that contains the itemset definition. An itemset is a group of data items that repeat. For example, a concomitant medication CRF might contain an itemset to capture the same set of data items about each concomitant medication the patient is using.
- *Item*—REFNAME of the item, as specified in the .xml file that contains the item definition.
- *Control*—REFNAME of the child control that contains the checkbox you want to test. To access a checkbox component of a group control, refer to each parent control in which the child component is nested. For example, to address one of two simple controls within a group control, enter the REFNAME of the group control followed by the REFNAME of the simple control, and separate the names with periods, as follows: *GroupControlRefname.SimpleControlRefname*.

**ItemsetIndex** Number that specifies an index into the items that make up an itemset. An itemset is a group of data items that repeat. For example, a concomitant medication CRF might contain an itemset to capture the same set of data items about each concomitant medication the patient is using. Valid ItemsetIndex values are 0 or greater:

- 0—Indicates that the control is not an itemset or, if the control is an itemset, references the current data item within the itemset. For example, if the user presses Submit after entering data in the third data item in the itemset, 0 sets the index to the third data item.
- Any number greater than 0—Explicitly indexes the specified item within the itemset.

**VisitIndex** Number that specifies an index into a repeating formset. A repeating formset is most often used for unscheduled visits in which the same data is recorded each time the visit occurs. Valid VisitIndex values are 0 or greater:

- 0—Indicates that the formset is not repeating or, if the formset is repeating, references the current formset.
- Any number greater than 0—Explicitly indexes the specified occurrence of the formset.

**FormIndex** Number that specifies an index into a repeating form.

- 0—References the current repeating form.
- 1—References a non-repeating form.
- Any number greater than 1—References the specified occurrence of the repeating form.

## Example

This example tests each check box in the Mthd\_GC control of the current instance of the CM form to determine whether it has been checked. If no check boxes have been checked, the rule issues a query.

```
patient.RulePassed=1
If patient.GetValueRF("0.CommonCRF.CM.sctCM.0.itmCMMed.txtCMMed", "", 0, 0, 0) <> "" then
If ("True" = patient.IsCBSelectedRF
("0.CommonCRF.CM.sctCM.0.itmMthd.Mthd_GC.Mthd_1_CBC.l_SC", 0, 0, 0)) or _
("True" = patient.IsCBSelectedRF
("0.CommonCRF.CM.sctCM.0.itmMthd.Mthd_GC.Mthd_i_CBC.i_SC", 0, 0, 0)) or _
("True" = patient.IsCBSelectedRF
("0.CommonCRF.CM.sctCM.0.itmMthd.Mthd_GC.Mthd_t_CBC.t_SC", 0, 0, 0)) then
Else
patient.RulePassed=0
patient.AddNamedValue "QUERYTEXT", "Medication is present but at least 1 method has not
been selected in item 6. Please select at least one checkbox in item 6."
End If
End If
```



## IsItemSetDeleted(REFNAMEPath,VisitIndex,ItemSetIndex)

The IsItemSetDeleted method indicates whether a row in an itemset is deleted by returning a zero if the row is not deleted, and a 1 if the row is deleted. If you are using a rule that checks for duplicate information within an itemset, and you have information that you have deleted, you do not want the rule to include the items in these deleted rows.

### Arguments

**REFNAMEPath** String that identifies the form component to get. Each REFNAME in the string is the REFNAME specified in the .xml file that contains the definition of the form component. The REFNAMEPath must be fully qualified; only the patient component of the path defaults to the current patient. The REFNAMEPath argument is in the following format:

(0.Visit.Form.Section.Itemset.Item[control[control...]])

- *0*—Current patient.
- *Visit*—REFNAME of the visit, as specified in the .xml file that contains the visit definition.
- *Form*—REFNAME of the form, as specified in the .xml file that contains the form definition.
- *Section*—REFNAME of the section, as specified in the .xml file that contains the section definition.
- *Itemset*—REFNAME of the itemset, as specified in the .xml file that contains the itemset definition. An itemset is a group of data items that repeat. For example, a concomitant medication CRF might contain an itemset to capture the same set of data items about each concomitant medication the patient is using.
- *Item*—REFNAME of the item, as specified in the .xml file that contains the item definition.
- *Control*—REFNAME of the control, as specified in the .xml file that contains the control definition. To access an component of a group control, refer to each parent control in which the child component is nested. For example, to address one of two text controls within a group control, enter the REFNAME of the group control followed by the REFNAME of the text control, and separate the names with periods, as follows: *GroupControlRefname.TextControlRefname*.

**VisitIndex** Number that specifies an index into a repeating formset. A repeating formset is most often used for unscheduled visits in which the same data is recorded each time the visit occurs. Valid VisitIndex values are 0 or greater:

- *0*—Indicates that the formset is not repeating or, if the formset is repeating, references the current formset.
- Any number greater than 0—Explicitly indexes the specified occurrence of the formset.

**ItemSetIndex** Number that specifies an index into the items that make up an itemset. An itemset is a group of data items that repeat. For example, a concomitant medication CRF might contain an itemset to capture the same set of data items about each concomitant medication the patient is using. Valid ItemSetIndex values are 0 or greater:

- *0*—Indicates that the control is not an itemset or, if the control is an itemset, references the current data item within the itemset. For example, if the user presses Submit after entering data in the third data item in the itemset, 0 sets the index to the third data item.
- Any number greater than 0—Explicitly indexes the specified item within the itemset.

## Example

```

if (Patient.IsItemSetDeleted(Patient.GetCurPath(),0,0)) then
 Result.RulePassed = 0
else
 Result.RulePassed = 1
End if

```

## IsItemSetDeletedRF(REFNAMEPath,VisitIndex,FormIndex,ItemsetIndex)

In an instance of a repeating form, the IsItemSetDeletedRF method indicates whether a row in an itemset is deleted by returning a zero if the row is not deleted, and a 1 if the row is deleted. If you are using a rule that checks for duplicate information within an itemset, and you have information that you have deleted, you do not want the rule to include the items in these deleted rows.

The IsItemSetDeletedRF method is identical to the IsItemSetDeleted method, except that it is intended for use in repeating forms. Therefore, it has an additional argument, FormIndex, which enables you to reference a specific instance of the repeating form.

## Arguments

REFNAMEPath String that identifies the form component to get. Each REFNAME in the string is the REFNAME specified in the .xml file that contains the definition of the form component. The REFNAMEPath must be fully qualified; only the patient component of the path defaults to the current patient. The REFNAMEPath argument is in the following format:  
(0.Visit.Form.Section.Itemset.Item[.control[.control...]])

- *0*—Current patient.
- *Visit*—REFNAME of the visit, as specified in the .xml file that contains the visit definition.
- *Form*—REFNAME of the form, as specified in the .xml file that contains the form definition.
- *Section*—REFNAME of the section, as specified in the .xml file that contains the section definition.
- *Itemset*—REFNAME of the itemset, as specified in the .xml file that contains the itemset definition. An itemset is a group of data items that repeat. For example, a concomitant medication CRF might contain an itemset to capture the same set of data items about each concomitant medication the patient is using.
- *Item*—REFNAME of the item, as specified in the .xml file that contains the item definition.
- *Control*—REFNAME of the control, as specified in the .xml file that contains the control definition. To access an component of a group control, refer to each parent control in which the child component is nested. For example, to address one of two text controls within a group control, enter the REFNAME of the group control followed by the REFNAME of the text control, and separate the names with periods, as follows: *GroupControlRefname.TextControlRefname*.

**VisitIndex** Number that specifies an index into a repeating formset. A repeating formset is most often used for unscheduled visits in which the same data is recorded each time the visit occurs. Valid VisitIndex values are 0 or greater:

- 0—Indicates that the formset is not repeating or, if the formset is repeating, references the current formset.
- Any number greater than 0—Explicitly indexes the specified occurrence of the formset.

**FormIndex** Number that specifies an index into a repeating form.

- 0—References the current repeating form.
- 1—References a non-repeating form.
- Any number greater than 1—References the specified occurrence of the repeating form.

**ItemsetIndex** Number that specifies an index into the items that make up an itemset. An itemset is a group of data items that repeat. For example, a concomitant medication CRF might contain an itemset to capture the same set of data items about each concomitant medication the patient is using. Valid ItemsetIndex values are 0 or greater:

- 0—Indicates that the control is not an itemset or, if the control is an itemset, references the current data item within the itemset. For example, if the user presses Submit after entering data in the third data item in the itemset, 0 sets the index to the third data item.
- Any number greater than 0—Explicitly indexes the specified item within the itemset.

## Example

This example checks whether the current itemset row in a repeating form instance has been deleted.

```
if (Patient.IsItemSetDeletedRF(Patient.GetCurPath(),0,0,0)) then
 Result.RulePassed = 0
else
 Result.RulePassed = 1
End if
```

## IsValidItemPath(RefNamePath,VisitIndex)

The IsValidItemPath method confirms whether a RefName path is valid, by checking the database for the existence of the specified path. When sites are on different trial versions of a trial and you want a calculation to update an item that does not exist in all trial versions, you can use this method to check whether the item exists before performing the calculation. Additionally, you can use the method to determine whether a calculation should update a new version of a form or an alternate form used to capture the same data for patients who started the form in an earlier trial version. The method returns 1 (True), indicating that the path exists, or 0 (False), indicating that the path does not exist.

**Note:** Use IsValidItemPath only to check for the existence of the path to a calculated item when you want to update the item with a calculation; do not use the method for form rules.

## Arguments

**RefNamePath** String that identifies the simple control of the component to be tested. Each REFNAME in the string is the REFNAME specified in the .xml file that contains the definition of the form component. The REFNAMEPath must be fully qualified; only the patient component of the path defaults to the current patient. The REFNAMEPath argument is in the following format: *(0.Visit.Form.Section.Itemset.Item[.control[.control...]])*

- *0*—Current patient.
- *Visit*—REFNAME of the visit, as specified in the .xml file that contains the visit definition.
- *Form*—REFNAME of the form, as specified in the .xml file that contains the form definition.
- *Section*—REFNAME of the section, as specified in the .xml file that contains the section definition.
- *Itemset*—REFNAME of the itemset, as specified in the .xml file that contains the itemset definition. An itemset is a group of data items that repeat. For example, a concomitant medication CRF might contain an itemset to capture the same set of data items about each concomitant medication the patient is using.
- *Item*—REFNAME of the item, as specified in the .xml file that contains the item definition.
- *Control*—REFNAME of the child control whose existence you want to test. To access a component of a group control, refer to each parent control in which the child component is nested. For example, to address one of two text controls within a group control, enter the REFNAME of the group control followed by the REFNAME of the text control, and separate the names with periods, as follows: *GroupControlRefname.SimpleControlRefname*.

**VisitIndex** Number that specifies an index into a repeating formset. A repeating formset is most often used for unscheduled visits in which the same data is recorded each time the visit occurs. Valid VisitIndex values are 0 or greater:

- *0*—Indicates that the formset is not repeating or, if the formset is repeating, references the current formset.
- Any number greater than 0—Explicitly indexes the specified occurrence of the formset.

## Example

In this example, if an item exists on the current version of a form, a calculated value is updated on that version; otherwise, the value is updated on an alternate form.

```
dpath=0.Visit1.DEM.DEM.0.NEWITEM.NEWITEM_TB
altdpath=0.Visit1.ALTDEM.ALTDEM.0.NEWITEM.NEWITEM_TB
val="Patient.GetEnteredValue("0.Visit1.DEM.DEM.0.WEIGHT.WEIGHTTEXT",0)"
newval=val * .625
```

```
If Patient.IsValidItemPath("dpath,0") then
```

```
 Patient.SetEnteredValue dpath,"", 0, 0, 0,newval
```

```
else Patient.SetEnteredValue altdpath,"", 0, 0, 0,newval
```

```
end if
```

## IsValidItemPathRF(RefNamePath,Visit,Index,FormIndex)

In instances of a repeating form, the IsValidItemPathRF method confirms whether a RefName path is valid, by checking the database for the existence of the specified path. When sites are on different trial versions of a trial and you want a calculation to update an item that does not exist in all trial versions, you can use this method to check whether the item exists before performing the calculation. Additionally, you can use the method to determine whether a calculation should update a new version of a form or an alternate form used to capture the same data for patients who started the form in an earlier trial version. The method returns 1 (True), indicating that the path exists, or 0 (False), indicating that the path does not exist.

The IsValidItemPathRF method is identical to the *IsValidItemPath* (see "*IsValidItemPath(RefNamePath, VisitIndex)*" on page 291) method, except that it is intended for use in repeating forms. Therefore, it has an additional argument, FormIndex, which enables you to reference a specific instance of the repeating form.

**Note:** Use IsValidItemPathRF only to check for the existence of the path to a calculated item when you want to update the item with a calculation; do not use the method for form rules.

## Arguments

**RefNamePath** String that identifies the simple control of the component to be tested. Each REFNAME in the string is the REFNAME specified in the .xml file that contains the definition of the form component. The REFNAMEPath must be fully qualified; only the patient component of the path defaults to the current patient. The REFNAMEPath argument is in the following format: *(0.Visit.Form.Section.Itemset.Item[.control[.control...]])*

- *0*—Current patient.
- *Visit*—REFNAME of the visit, as specified in the .xml file that contains the visit definition.
- *Form*—REFNAME of the form, as specified in the .xml file that contains the form definition.
- *Section*—REFNAME of the section, as specified in the .xml file that contains the section definition.
- *Itemset*—REFNAME of the itemset, as specified in the .xml file that contains the itemset definition. An itemset is a group of data items that repeat. For example, a concomitant medication CRF might contain an itemset to capture the same set of data items about each concomitant medication the patient is using.
- *Item*—REFNAME of the item, as specified in the .xml file that contains the item definition.
- *Control*—REFNAME of the child control whose existence you want to test. To access a component of a group control, refer to each parent control in which the child component is nested. For example, to address one of two text controls within a group control, enter the REFNAME of the group control followed by the REFNAME of the text control, and separate the names with periods, as follows: *GroupControlRefname.SimpleControlRefname*.

**VisitIndex** Number that specifies an index into a repeating formset. A repeating formset is most often used for unscheduled visits in which the same data is recorded each time the visit occurs. Valid VisitIndex values are 0 or greater:

- *0*—Indicates that the formset is not repeating or, if the formset is repeating, references the current formset.
- Any number greater than *0*—Explicitly indexes the specified occurrence of the formset.

**FormIndex** Number that specifies an index into a repeating form.

- *0*—References the current repeating form.
- *1*—References a non-repeating form.
- Any number greater than *1*—References the specified occurrence of the repeating form.

## Example

In this example, if an item exists on the current version of a form, a calculated value is updated on that version; otherwise, the value is updated on an alternate form.

```
dpath=0.vstCORE2.0.frmECG.sctECG.0.mitmQTRate.calQTRate
```

```
altdpath=0.vstCORE2.0.frmALTECG.sctALTECG.0.NEWitmQTRate.NEWcalRate
```

```
val="Patient.GetEnteredValueRF("0.vstCORE2.0.frmALTECG.sctALTECG.0.mitmQRSRate.mtxtQRSRate",0")
```

```
newval=val * .625
```

```
If Patient.IsValidItemPathRF("dpath,0,0") then
 Patient.SetEnteredValueRF dpath,"",0,0,0,0,newval
else Patient.SetEnteredValueRF altdpath,"",0,0,0,0,newval
end if
```

## ItemHasBeenNormalizedEx(REFNAMEPath,ItemsetIndex,VisitIndex)

The `ItemHasBeenNormalizedEx` method indicates whether the units associated with the referenced data item have been selected, by determining whether the data has been normalized. When a user enters a data value and submits the data, the Id of the units conversion rule is passed to the rules engine if the user has selected the unit with which the rule is associated. If the rules engine gets the conversion rule Id, the `ItemHasBeenNormalizedEx` method evaluates to true; otherwise, the method evaluates to false.

**Note:** To be able to use this method, you must ensure that each unit, including the base unit, is associated with a conversion rule. The conversion rule for the base unit should be `Data.Result=Data.BaseValue*1`.

## Arguments

REFNAMEPath String that identifies the form component to get. Each REFNAME in the string is the REFNAME specified in the .xml file that contains the definition of the form component. The REFNAMEPath must be fully qualified; only the patient component of the path defaults to the current patient. The REFNAMEPath argument is in the following format:  
(0.Visit.Form.Section.Itemset.Item[.control[.control...]])

- *0*—Current patient.
- *Visit*—REFNAME of the visit, as specified in the .xml file that contains the visit definition.
- *Form*—REFNAME of the form, as specified in the .xml file that contains the form definition.
- *Section*—REFNAME of the section, as specified in the .xml file that contains the section definition.
- *Itemset*—REFNAME of the itemset, as specified in the .xml file that contains the itemset definition. An itemset is a group of data items that repeat. For example, a concomitant medication CRF might contain an itemset to capture the same set of data items about each concomitant medication the patient is using.
- *Item*—REFNAME of the item, as specified in the .xml file that contains the item definition.
- *Control*—REFNAME of the control, as specified in the .xml file that contains the control definition. To access an component of a group control, refer to each parent control in which the child component is nested. For example, to address one of two text controls within a group control, enter the REFNAME of the group control followed by the REFNAME of the text control, and separate the names with periods, as follows: *GroupControlRefname.TextControlRefname*.

**ItemSetIndex** Number that specifies an index into the items that make up an itemset. An itemset is a group of data items that repeat. For example, a concomitant medication CRF might contain an itemset to capture the same set of data items about each concomitant medication the patient is using. Valid ItemSetIndex values are 0 or greater:

- 0—Indicates that the control is not an itemset or, if the control is an itemset, references the current data item within the itemset. For example, if the user presses Submit after entering data in the third data item in the itemset, 0 sets the index to the third data item.
- Any number greater than 0—Explicitly indexes the specified item within the itemset.

**VisitIndex** Number that specifies an index into a repeating formset. A repeating formset is most often used for unscheduled visits in which the same data is recorded each time the visit occurs. Valid VisitIndex values are 0 or greater:

- 0—Indicates that the formset is not repeating or, if the formset is repeating, references the current formset.
- Any number greater than 0—Explicitly indexes the specified occurrence of the formset.

## Example

```
UnitFilled = Patient.ItemHasBeenNormalizedEx("0.Visit1.DEM.DEM.0.HEIGHT",0,0)
```

## ItemHasBeenNormalizedRF(REFNAMEPath,ItemsetIndex,VisitIndex,FormIndex)

For an item occurring in a specific instance of a repeating form, the ItemHasBeenNormalizedRF method indicates whether the units associated with the referenced data item have been selected, by determining whether the data has been normalized. When a user enters a data value and submits the data, the Id of the units conversion rule is passed to the rules engine if the user has selected the unit with which the rule is associated. If the rules engine gets the conversion rule Id, the ItemHasBeenNormalizedRF method evaluates to true; otherwise, the method evaluates to false.

The ItemHasBeenNormalizedRF method is identical to the *ItemHasBeenNormalizedEx* (see "*ItemHasBeenNormalizedEx(REFNAMEPath,ItemsetIndex,VisitIndex)*" on page 295) method, except that it is intended for use in repeating forms. Therefore, it has an additional argument, FormIndex, which enables you to reference a specific instance of the repeating form.

**Note:** To be able to use this method, you must ensure that each unit, including the base unit, is associated with a conversion rule. The conversion rule for the base unit should be `Data.Result=Data.BaseValue*1`.



## Arguments

REFNAMEPath String that identifies the form component to get. Each REFNAME in the string is the REFNAME specified in the .xml file that contains the definition of the form component. The REFNAMEPath must be fully qualified; only the patient component of the path defaults to the current patient. The REFNAMEPath argument is in the following format:

(0.Visit.Form.Section.Itemset.Item[.control[.control...]])

- 0—*Current patient.*
- *Visit*—REFNAME of the visit, as specified in the .xml file that contains the visit definition.
- *Form*—REFNAME of the form, as specified in the .xml file that contains the form definition.
- *Section*—REFNAME of the section, as specified in the .xml file that contains the section definition.
- *Itemset*—REFNAME of the itemset, as specified in the .xml file that contains the itemset definition. An itemset is a group of data items that repeat. For example, a concomitant medication CRF might contain an itemset to capture the same set of data items about each concomitant medication the patient is using.
- *Item*—REFNAME of the item, as specified in the .xml file that contains the item definition.
- *Control*—REFNAME of the control, as specified in the .xml file that contains the control definition. To access an component of a group control, refer to each parent control in which the child component is nested. For example, to address one of two text controls within a group control, enter the REFNAME of the group control followed by the REFNAME of the text control, and separate the names with periods, as follows: *GroupControlRefname.TextControlRefname.*

**ItemSetIndex** Number that specifies an index into the items that make up an itemset. An itemset is a group of data items that repeat. For example, a concomitant medication CRF might contain an itemset to capture the same set of data items about each concomitant medication the patient is using. Valid ItemSetIndex values are 0 or greater:

- 0—Indicates that the control is not an itemset or, if the control is an itemset, references the current data item within the itemset. For example, if the user presses Submit after entering data in the third data item in the itemset, 0 sets the index to the third data item.
- Any number greater than 0—Explicitly indexes the specified item within the itemset.

**VisitIndex** Number that specifies an index into a repeating formset. A repeating formset is most often used for unscheduled visits in which the same data is recorded each time the visit occurs. Valid VisitIndex values are 0 or greater:

- 0—Indicates that the formset is not repeating or, if the formset is repeating, references the current formset.
- Any number greater than 0—Explicitly indexes the specified occurrence of the formset.

**FormIndex** Number that specifies an index into a repeating form.

- 0—References the current repeating form.
- 1—References a non-repeating form.
- Any number greater than 1—References the specified occurrence of the repeating form.

## Example

This example illustrates the use of `ItemHasBeenNormalized` to check whether a control in the current instance of a repeating form has been selected.

```
IsSelected =
Patient.ItemHasBeenNormalizedRF(Patient.GetCurPath(),0,0,Patient.GetCurrentFormIndex())
patient.rulepassed = 1
Patient.AddNamedValue "QUERYTEXT", "Quantity is present, but measurements are not selected;
please complete"
if IsSelected = 0 then
patient.rulepassed = 0
end if
```

## LoadCRB()

The `LoadCRB` method loads all of the data in the current patient's Case Book into memory. This provides better performance when a rule must check multiple data items in several visits and forms, as the rule can find data without querying the database multiple times.

## OutputDebug(PrintString, Variable)

The `OutputDebug` method displays the value of a specified variable in the InForm Architect application Output window as a test script runs.

It is not necessary to remove this before running the InForm application, as this method is non-operational in the InForm system.

## Arguments

**PrintString** String that identifies the value to trace.

**Variable**

Name of the variable for which to display the current value.

## Example

```
UnitFilled = Patient.ItemHasBeenNormalizedEx("0.Visit1.DEM.DEM.0.HEIGHT",0,0)
Patient.OutputDebug "TestUnit", UnitFilled
```

## RemoveEmptyDynamicVisit

The `RemoveEmptyDynamicVisit` method removes a dynamic visit that a patient has been scheduled into, if no data has been entered into any of the visit's forms. The method takes the `RefName` of the dynamic visit to remove as its only argument.

### Removing a dynamic visit

If no patient data has been entered in any form in a dynamic visit, you can remove the visit by calling the `RemoveEmptyDynamicVisit` method. `RemoveEmptyDynamicVisit` has a single parameter: the `RefName` of the dynamic visit.

### Example

In this example a patient is removed from an erroneously scheduled dynamic visit and rescheduled into the appropriate visit for the patient's gender.

```

MALE=1
FEMALE=2
MALEVISIT="Dose100"
FEMALEVISIT="Dose200"
Gender = Patient.GetValue("0.Visit1.DEM.DEM.0.GENDER","",0,0,0)

If Gender = MALE Then
 SVisit="MALEVISIT"
 RVisit="FEMALEVISIT"
Else
 SVisit="FEMALEVISIT"
 RVisit="MALEVISIT"
End If
Patient.RemoveEmptyDynamicVisit RVisit
Patient.SetDynamicVisitStart SVisit, "", 720

```

## RemoveNotStartedDynamicForm

If no data has been entered into a dynamic form, you can remove it from the patient's Case Book by using the `RemoveNotStartedDynamicForm` method on the `Patient` or `Result` object.

`RemoveNotStatedDynamicForm` has the following parameters:

- `RefName` of the visit in which to the form appears or may appear.
- `RefName` of the dynamic form to remove from the schedule.
- `Visit Index`. This is a number that specifies an index into a repeating visit. Values are:
  - 0—Indicates that the visit is not repeating or, if the visit is repeating, references the current instance of the visit.
  - Any number greater than 0—Explicitly indexes the specified occurrence of the visit.

## Example

```
MALE=1
FEMALE=2
```

```
Gender=Patient.GetValue("0.visit1.DEM.DEM.0.GENDER", "", 0, 0, 0)
If Gender=FEMALE Then
Patient.ShowDynamicForm "visit1" , "Preg", 0
Else
Patient.RemoveNotStartedDynamicForm "visit1", "Preg", 0
End If
```

The "Preg" dynamic form appears in the visit if the gender of the patient is female. If the gender is not female, the form is removed from the visit and will not be scheduled for the patient.

## SetDynamicVisitStart

The SetDynamicVisitStart method schedules a patient into a dynamic visit. The method takes the following arguments: RefName of the dynamic visit to schedule, RefName of the visit that precedes the dynamic visit, and the number of hours from the preceding visit that the dynamic visit starts. If the preceding visit is not specified, the hours are counted from the first visit of the trial.

To assign a patient to a dynamic visit schedule, create a calculation rule script that tests the value of the item on which the visit assignment is based and makes the assignment. Use the SetDynamicVisitStart method on the Patient or Result object.

SetDynamicVisitStart has the following parameters:

- RefName of the dynamic visit to schedule.
- RefName of the visit preceding the dynamic visit.
- Number of hours before the dynamic visit starts. If you specify a preceding visit RefName, this is the number of hours from that visit. If you do not specify a preceding visit RefName, this is the number of hours from the beginning of the trial.

## Example 1

In this example, the patient's gender determines whether the patient is scheduled for the Dose100 or the Dose250 visit, both of which start 720 hours after the patient's first visit.

```
MALE=1
FEMALE=2
```

```
Gender = Patient.GetValue("0.Visit1.DEM.DEM.0.GENDER", "", 0, 0, 0)
```

```
If Gender = MALE Then
 DVisit="Dose100"
Else
 DVisit="Dose250"
End If
```

```
Patient.SetDynamicVisitStart DVisit, "", 720
```

## Example 2

In this example the patient's gender determines whether the patient starts the Visit4Gen dynamic visit ten days or two weeks after the previous visit, Visit3.

MALE=1

FEMALE=2

Gender = Patient.GetValue("0.Visit1.DEM.DEM.0.GENDER","",0,0,0)

If Gender = MALE Then

    StartAt=240

Else

    StartAt=336

End If

Patient.SetDynamicVisitStart "Visit4Gen", "Visit3", StartAt

## SetEnteredValue(REFNAMEPath,AttributeName,Index,ItemsetIndex,VisitIndex,Value)

The SetEnteredValue method uses the value of the data entered in the specified form component. SetEnteredValue is very similar to the *SetValue method* (see "*SetValue(REFNAMEPath,AttributeName,Index,ItemsetIndex,VisitIndex,Value)*" on page 312) method, which uses the normalized value of the data entered in the specified form component. The difference between these methods applies to form components that are defined as units.

The MedML Installer utility enables unit form components to be defined with reference to other units, so that data can be displayed online as one unit and stored in the database, after conversion, as the other. For example, you can design a form so that a CRA can choose whether to enter a weight in ounces or pounds, and specify that the base unit is ounces. The InForm application stores the value as entered, whether it is ounces or pounds. If the entered value is pounds, the InForm application converts it to ounces, using a conversion rule that is attached to the unit definition, and stores the value in ounces in the database. The unit in which data is stored in the database after conversion is called the base unit, and the data value, after conversion to the base unit, is called the normalized value.

When you want to use the entered value of a data item, use SetEnteredValue. When you must use the normalized value of a data item, use SetValue.

**Note:** Use SetEnteredValue only in calculations, not in form rules.

## Arguments

**REFNAMEPath** String that identifies the form component to use. Each REFNAME in the string is the REFNAME specified in the .xml file that contains the definition of the form component. The REFNAMEPath must be fully qualified; only the patient component of the path defaults to the current patient. The REFNAMEPath argument is in the following format:

(0.Visit.Form.Section.Itemset.Item[.control[.control...]])

- *0*—Current patient.
- *Visit*—REFNAME of the visit, as specified in the .xml file that contains the visit definition.
- *Form*—REFNAME of the form, as specified in the .xml file that contains the form definition.
- *Section*—REFNAME of the section, as specified in the .xml file that contains the section definition.
- *Itemset*—REFNAME of the itemset, as specified in the .xml file that contains the itemset definition. An itemset is a group of data items that repeat. For example, a concomitant medication CRF might contain an itemset to capture the same set of data items about each concomitant medication the patient is using.
- *Item*—REFNAME of the item, as specified in the .xml file that contains the item definition.
- *Control*—REFNAME of the control, as specified in the .xml file that contains the control definition. To access an component of a group control, refer to each parent control in which the child component is nested. For example, to address one of two text controls within a group control, enter the REFNAME of the group control followed by the REFNAME of the text control and separated by periods, as follows: *GroupControlRefname.TextControlRefname*.

**AttributeName** String that specifies the type of information to return, specified in double quotes ("").

**Note:** When you use this method, do not use double quotes (") to indicate an empty string in the AttributeName if the control is in a required item. Use EMPTY instead. Otherwise, the item will not be yellow, and there will be no indication that required information is missing.

Valid AttributeNames are:

- **VALUE**—Gets the value of the form component. This attribute is the default; you can indicate it with empty double quotes unless the control using this method is used in a required item.
- **VALIDDATEMAP** — Number that specifies whether a date is complete. The script uses the VALIDDATEMAP attribute of the GetValue method on the Patient object. VALIDDATEMAP returns an integer that contains a bitmask indicating which elements of the DateTimeControl are complete:
  - **Year**—1
  - **Month**—2
  - **Day**—4
  - **Hour**—8
  - **Minute**—16
  - **Second**—32

- **UNKNOWNDATEMAP** — Number that specifies whether a date has components that are marked Unknown. The script uses the UNKNOWNDATEMAP attribute of the GetValue method on the Patient object. UNKNOWNDATEMAP returns an integer that contains a bitmask indicating which elements of the DateTimeControl are unknown:
  - **Year**—1
  - **Month**—2
  - **Day**—4
  - **Hour**—8
  - **Minute**—16
  - **Second**—32

**Note:** If you set a bit in the UNKNOWNDATEMAP value for a date component, the corresponding bit is not set in the VALIDDATEMAP value.

### Index

Number, starting with 0, that specifies an index into an array of the selected items in a multiple-value control such as a list of check boxes. To use the Index attribute, either first return the COUNT of the control to determine how many items are selected, or find the selected items by looping through the list and creating an array of the selected items.

### ItemSetIndex

Number that specifies an index into the items that make up an itemset. An itemset is a group of data items that repeat. For example, a concomitant medication CRF might contain an itemset to capture the same set of data items about each concomitant medication the patient is using. Valid ItemSetIndex values are 0 or greater:

- 0—Indicates that the control is not an itemset or, if the control is an itemset, references the current data item within the itemset. For example, if the user presses Submit after entering data in the third data item in the itemset, 0 sets the index to the third data item.
- Any number greater than 0—Explicitly indexes the specified item within the itemset.

**VisitIndex** Number that specifies an index into a repeating formset. A repeating formset is most often used for unscheduled visits in which the same data is recorded each time the visit occurs. Valid VisitIndex values are 0 or greater:

- 0—Indicates that the formset is not repeating or, if the formset is repeating, references the current formset.
- Any number greater than 0—Explicitly indexes the specified occurrence of the formset.

**Value** Value to set.

## Example

In the following example, the patient ID is added to a site number retrieved by SetEnteredValue  
 v2subno="0.V2.SUBID.SUBID\_S.0.SUBNO"  
 v2scrno="0.V2.SUBID.SUBID\_S.0.SCNUM"

```
pn1 = cstr(Patient.GetValue(v2subno, "", 0, 0, 0))
```

```
sitenm_g=Patient.GetSiteName()
siteno = cSTR(mid(sitenm_g,2,2))
```

```
If isnumeric(pn1) and len(pn1)=2 then
```

```
 Patient.SetEnteredValue v2subno,"", 0, 0, 0,siteno&pn1
```

```
else if len(pn1)=3 and cint(pn1)>600 and cint(pn1)<700 then
```

```
 Patient.SetEnteredValue v2subno,"",0,0,0,"S"&siteno&pn1
```

```
 Patient.SetEnteredValue v2scrno,"",0,0,0,"S"&siteno&pn1
```

```
end if
```

```
end if
```

## SetEnteredValueRF(REFNAMEPath,AttributeName, Index, ItemsetIndex, VisitIndex, FormIndex, Value)

In instances of a repeating form, the SetEnteredValueRF method uses the value of the data entered in the specified form component. The SetEnteredValueRF method is identical to the *SetEnteredValue* (see

*"SetEnteredValue(REFNAMEPath,AttributeName,Index,ItemsetIndex,VisitIndex,Value)"* on page 301) method, except that it is intended for use in repeating forms. Therefore, it has an additional argument, FormIndex, which enables you to reference a specific instance of the repeating form.

**Note:** Use SetEnteredValueRF only in calculations, not in form rules.



## Arguments

**REFNAMEPath** String that identifies the form component to use. Each REFNAME in the string is the REFNAME specified in the .xml file that contains the definition of the form component. The REFNAMEPath must be fully qualified; only the patient component of the path defaults to the current patient. The REFNAMEPath argument is in the following format:  
(0.Visit.Form.Section.Itemset.Item[.control[.control...]])

- *0*—Current patient.
- *Visit*—REFNAME of the visit, as specified in the .xml file that contains the visit definition.
- *Form*—REFNAME of the form, as specified in the .xml file that contains the form definition.
- *Section*—REFNAME of the section, as specified in the .xml file that contains the section definition.
- *Itemset*—REFNAME of the itemset, as specified in the .xml file that contains the itemset definition. An itemset is a group of data items that repeat. For example, a concomitant medication CRF might contain an itemset to capture the same set of data items about each concomitant medication the patient is using.
- *Item*—REFNAME of the item, as specified in the .xml file that contains the item definition.
- *Control*—REFNAME of the control, as specified in the .xml file that contains the control definition. To access an component of a group control, refer to each parent control in which the child component is nested. For example, to address one of two text controls within a group control, enter the REFNAME of the group control followed by the REFNAME of the text control and separated by periods, as follows: *GroupControlRefname.TextControlRefname*.

**AttributeName** String that specifies the type of information to return, specified in double quotes (""). Valid AttributeNames are:

- **VALUE**—Gets the value of the form component. This attribute is the default; you can indicate it with empty double quotes.
- **VALIDDATEMAP**—Number that specifies whether a date is complete. The script uses the VALIDDATEMAP attribute of the SetEnteredValue method on the Patient object. VALIDDATEMAP returns an integer that contains a bitmask indicating which elements of the DateTimeControl are complete:

Year	1
Month	2
Day	4
Hour	8
Minute	16
Second	32

- **UNKNOWNDATEMAP**—Number that specifies whether a date has components that are marked Unknown. The script uses the UNKNOWNDATEMAP attribute of the SetEnteredValue method on the Patient object. UNKNOWNDATEMAP returns an integer that contains a bitmask indicating which elements of the DateTimeControl are unknown:

Year	1
Month	2
Day	4
Hour	8
Minute	16
Second	32

### Index

Number, starting with 0, that specifies an index into an array of the selected items in a multiple-value control such as a list of check boxes. To use the Index attribute, either first return the COUNT of the control to determine how many items are selected, or find the selected items by looping through the list and creating an array of the selected items.

### ItemSetIndex

Number that specifies an index into the items that make up an itemset. An itemset is a group of data items that repeat. For example, a concomitant medication CRF might contain an itemset to capture the same set of data items about each concomitant medication the patient is using. Valid ItemSetIndex values are 0 or greater:

- 0—Indicates that the control is not an itemset or, if the control is an itemset, references the current data item within the itemset. For example, if the user presses Submit after entering data in the third data item in the itemset, 0 sets the index to the third data item.
- Any number greater than 0—Explicitly indexes the specified item within the itemset.

**VisitIndex** Number that specifies an index into a repeating formset. A repeating formset is most often used for unscheduled visits in which the same data is recorded each time the visit occurs. Valid VisitIndex values are 0 or greater:

- 0—Indicates that the formset is not repeating or, if the formset is repeating, references the current formset.
- Any number greater than 0—Explicitly indexes the specified occurrence of the formset.

**FormIndex** Number that specifies an index into a repeating form.

- 0—References the current repeating form.
- 1—References a non-repeating form.
- Any number greater than 1—References the specified occurrence of the repeating form.

**Value** Value to set.

## Example

In this example the value of a control identified by the `strSetBox2Path` variable is set to "\*\*\*\*\*"

```
strSetBox2Value = Patient.GetValueRF(strSetBox2Path,"",0,0,0,0)
If strSetBox2Value <> "" then
Patient.SetEnteredValueRF strSetBox2Path,"",0,0,0,0,"*****"
End If
```

## SetNewValue(REFNAMEPath,AttributeName,Index,ItemsetIndex,VisitIndex,Value)

The `SetNewValue` method is for internal Oracle use only. Like the `SetValue` method, it uses the value of the data entered in the specified form component. `SetNewValue` is only for items that have never had data entered in them. It is intended for use in the specialized enrollment mapping calculation that maps data from the Enrollment form to other forms in the trial.

**Note:** Do not use `SetNewValue` in a form rule.

## Arguments

`REFNAMEPath` String that identifies the form component to use. Each `REFNAME` in the string is the `REFNAME` specified in the .xml file that contains the definition of the form component. The `REFNAMEPath` must be fully qualified; only the patient component of the path defaults to the current patient. The `REFNAMEPath` argument is in the following format:  
(0.Visit.Form.Section.Itemset.Item[.control[.control...]])

- *0*—*Current patient.*
- *Visit*—*REFNAME* of the visit, as specified in the .xml file that contains the visit definition.
- *Form*—*REFNAME* of the form, as specified in the .xml file that contains the form definition.
- *Section*—*REFNAME* of the section, as specified in the .xml file that contains the section definition.
- *Itemset*—*REFNAME* of the itemset, as specified in the .xml file that contains the itemset definition. An itemset is a group of data items that repeat. For example, a concomitant medication CRF might contain an itemset to capture the same set of data items about each concomitant medication the patient is using.
- *Item*—*REFNAME* of the item, as specified in the .xml file that contains the item definition.
- *Control*—*REFNAME* of the control, as specified in the .xml file that contains the control definition. To access an component of a group control, refer to each parent control in which the child component is nested. For example, to address one of two text controls within a group control, enter the *REFNAME* of the group control followed by the *REFNAME* of the text control, and separate the names with periods, as follows: *GroupControlRefname.TextControlRefname.*

**AttributeName** String that specifies the type of information to use, specified in double quotes (""). Valid AttributeNames are:

- **VALUE**—Gets the value of the form component. This attribute is the default; you can indicate it with empty double quotes.
- **VALIDDATEMAP** — Number that specifies whether a date is complete. The script uses the **VALIDDATEMAP** attribute of the **GetValue** method on the **Patient** object. **VALIDDATEMAP** returns an integer that contains a bitmask indicating which elements of the **DateTimeControl** are complete:
  - **Year**—1
  - **Month**—2
  - **Day**—4
  - **Hour**—8
  - **Minute**—16
  - **Second**—32
- **UNKNOWNDATEMAP** — Number that specifies whether a date has components that are marked Unknown. The script uses the **UNKNOWNDATEMAP** attribute of the **GetValue** method on the **Patient** object. **UNKNOWNDATEMAP** returns an integer that contains a bitmask indicating which elements of the **DateTimeControl** are unknown:
  - **Year**—1
  - **Month**—2
  - **Day**—4
  - **Hour**—8
  - **Minute**—16
  - **Second**—32

**Note:** If you set a bit in the **UNKNOWNDATEMAP** value for a date component, the corresponding bit is not set in the **VALIDDATEMAP** value.

**Index**

Number, starting with 0, that specifies an index into an array of the selected items in a multiple-value control such as a list of check boxes. To use the Index attribute, either first return the COUNT of the control to determine how many items are selected, or find the selected items by looping through the list and creating an array of the selected items.

**ItemSetIndex**

Number that specifies an index into the items that make up an itemset. An itemset is a group of data items that repeat. For example, a concomitant medication CRF might contain an itemset to capture the same set of data items about each concomitant medication the patient is using. Valid ItemSetIndex values are 0 or greater:

- 0—Indicates that the control is not an itemset or, if the control is an itemset, references the current data item within the itemset. For example, if the user presses Submit after entering data in the third data item in the itemset, 0 sets the index to the third data item.
- Any number greater than 0—Explicitly indexes the specified item within the itemset.

**VisitIndex** Number that specifies an index into a repeating formset. A repeating formset is most often used for unscheduled visits in which the same data is recorded each time the visit occurs. Valid VisitIndex values are 0 or greater:

- 0—Indicates that the formset is not repeating or, if the formset is repeating, references the current formset.
- Any number greater than 0—Explicitly indexes the specified occurrence of the formset.

**Value** Value to set.

### **SetNewValueRF(REFNAMEPath,AttributeName,Index,ItemsetIndex,VisitIndex,FormIndex,Value)**

The SetNewValueRF method is for internal Oracle use only. Like the SetValueRF method, it uses the value of the data entered in the specified form component. SetNewValueRF is only for items that have never had data entered in them. It is intended for use in the specialized enrollment mapping calculation that maps data from the Enrollment form to other forms in the trial.

The SetNewValueRF method is identical to the SetNewValue method, except that it is intended for use in repeating forms. Therefore, it has an additional argument, FormIndex, which enables you to reference a specific instance of the repeating form.

**Note:** Do not use SetNewValueRF in a form rule.

## Arguments

REFNAMEPath String that identifies the form component to use. Each REFNAME in the string is the REFNAME specified in the .xml file that contains the definition of the form component. The REFNAMEPath must be fully qualified; only the patient component of the path defaults to the current patient. The REFNAMEPath argument is in the following format:

(0.Visit.Form.Section.Itemset.Item[.control[.control...]])

- *0*—Current patient.
- *Visit*—REFNAME of the visit, as specified in the .xml file that contains the visit definition.
- *Form*—REFNAME of the form, as specified in the .xml file that contains the form definition.
- *Section*—REFNAME of the section, as specified in the .xml file that contains the section definition.
- *Itemset*—REFNAME of the itemset, as specified in the .xml file that contains the itemset definition. An itemset is a group of data items that repeat. For example, a concomitant medication CRF might contain an itemset to capture the same set of data items about each concomitant medication the patient is using.
- *Item*—REFNAME of the item, as specified in the .xml file that contains the item definition.
- *Control*—REFNAME of the control, as specified in the .xml file that contains the control definition. To access an component of a group control, refer to each parent control in which the child component is nested. For example, to address one of two text controls within a group control, enter the REFNAME of the group control followed by the REFNAME of the text control, and separate the names with periods, as follows: *GroupControlRefname.TextControlRefname*.

**AttributeName** String that specifies the type of information to use, specified in double quotes (""). Valid AttributeNames are:

- **VALUE**—Gets the value of the form component. This attribute is the default; you can indicate it with empty double quotes.
- **VALIDDATEMAP** — Number that specifies whether a date is complete. The script uses the VALIDDATEMAP attribute of the GetValue method on the Patient object. VALIDDATEMAP returns an integer that contains a bitmask indicating which elements of the DateTimeControl are complete:
  - **Year**—1
  - **Month**—2
  - **Day**—4
  - **Hour**—8
  - **Minute**—16
  - **Second**—32
- **UNKNOWNDATEMAP** — Number that specifies whether a date has components that are marked Unknown. The script uses the UNKNOWNDATEMAP attribute of the GetValue method on the Patient object. UNKNOWNDATEMAP returns an integer that contains a bitmask indicating which elements of the DateTimeControl are unknown:
  - **Year**—1

- **Month**—2
- **Day**—4
- **Hour**—8
- **Minute**—16
- **Second**—32

**Note:** If you set a bit in the UNKNOWNDATEMAP value for a date component, the corresponding bit is not set in the VALIDDATEMAP value.

### Index

Number, starting with 0, that specifies an index into an array of the selected items in a multiple-value control such as a list of check boxes. To use the Index attribute, either first return the COUNT of the control to determine how many items are selected, or find the selected items by looping through the list and creating an array of the selected items.

### ItemSetIndex

Number that specifies an index into the items that make up an itemset. An itemset is a group of data items that repeat. For example, a concomitant medication CRF might contain an itemset to capture the same set of data items about each concomitant medication the patient is using. Valid ItemSetIndex values are 0 or greater:

- 0—Indicates that the control is not an itemset or, if the control is an itemset, references the current data item within the itemset. For example, if the user presses Submit after entering data in the third data item in the itemset, 0 sets the index to the third data item.
- Any number greater than 0—Explicitly indexes the specified item within the itemset.

**VisitIndex** Number that specifies an index into a repeating formset. A repeating formset is most often used for unscheduled visits in which the same data is recorded each time the visit occurs. Valid VisitIndex values are 0 or greater:

- 0—Indicates that the formset is not repeating or, if the formset is repeating, references the current formset.
- Any number greater than 0—Explicitly indexes the specified occurrence of the formset.

**FormIndex** Number that specifies an index into a repeating form.

- 0—References the current repeating form.
- 1—References a non-repeating form.
- Any number greater than 1—References the specified occurrence of the repeating form.

**Value** Value to set.

## Example

This example sets a value in a previously unentered repeating form to the value of a control in another form.

```
Sub CopyDateValue(strGetPath, strSetPath)
x = Patient.GetValueRF(strGetPath, "", 0, 0, 0, 0)
y = CStr(x)
If y<>"" Then Patient.SetNewValueRF strSetPath, "", 0, 0, 0, x
End Sub
```

## SetValue(REFNAMEPath,AttributeName,Index,ItemsetIndex,VisitIndex,Value)

The SetValue method uses the normalized value of the data entered in the specified form component. SetValue is very similar to the *SetEnteredValue* (see "*SetEnteredValue(REFNAMEPath,AttributeName,Index,ItemsetIndex,VisitIndex,Value)*" on page 301) method, which uses the entered value of the data entered in the specified form component. The difference between these methods applies to form components that are defined as units.

The MedML Installer utility enables unit form components to be defined with reference to other units, so that data can be displayed online as one unit and stored in the database, after conversion, as the other. For example, you can design a form so that a CRA can choose whether to enter a weight in ounces or pounds, and specify that the base unit is ounces. The InForm application stores the value as entered, whether it is ounces or pounds. If the entered value is pounds, the InForm application converts it to ounces, using a conversion rule that you attach to the unit definition, and stores the value in ounces in the database. The unit in which data is stored in the database after conversion is called the base unit, and the data value, after conversion to the base unit, is called the normalized value.

When you want to use a data value as entered, use SetEnteredValue. When you must use the normalized value of a data item, use SetValue.

**Note:** Use SetValue only in calculations, not in form rules.

## Arguments

REFNAMEPath String that identifies the form component to use. Each REFNAME in the string is the REFNAME specified in the .xml file that contains the definition of the form component. The REFNAMEPath must be fully qualified; only the patient component of the path defaults to the current patient. The REFNAMEPath argument is in the following format:  
(0.Visit.Form.Section.Itemset.Item[control[control...]])

- *0*—Current patient.
- *Visit*—REFNAME of the visit, as specified in the .xml file that contains the visit definition.
- *Form*—REFNAME of the form, as specified in the .xml file that contains the form definition.
- *Section*—REFNAME of the section, as specified in the .xml file that contains the section definition.



- *Itemset*—REFNAME of the itemset, as specified in the .xml file that contains the itemset definition. An itemset is a group of data items that repeat. For example, a concomitant medication CRF might contain an itemset to capture the same set of data items about each concomitant medication the patient is using.
- *Item*—REFNAME of the item, as specified in the .xml file that contains the item definition.
- *Control*—REFNAME of the control, as specified in the .xml file that contains the control definition. To access an component of a group control, refer to each parent control in which the child component is nested. For example, to address one of two text controls within a group control, enter the REFNAME of the group control followed by the REFNAME of the text control, and separate the names with periods, as follows: *GroupControlRefname.TextControlRefname*.

**AttributeName** String that specifies the type of information to use, specified in double quotes (""). Valid AttributeNames are:

- **VALUE**—Gets the value of the form component. This attribute is the default; you can indicate it with empty double quotes.
- **VALIDDATEMAP** — Number that specifies whether a date is complete. The script uses the VALIDDATEMAP attribute of the GetValue method on the Patient object. VALIDDATEMAP returns an integer that contains a bitmask indicating which elements of the DateTimeControl are complete:
  - **Year**—1
  - **Month**—2
  - **Day**—4
  - **Hour**—8
  - **Minute**—16
  - **Second**—32
- **UNKNOWNDATEMAP** — Number that specifies whether a date has components that are marked Unknown. The script uses the UNKNOWNDATEMAP attribute of the GetValue method on the Patient object. UNKNOWNDATEMAP returns an integer that contains a bitmask indicating which elements of the DateTimeControl are unknown:
  - **Year**—1
  - **Month**—2
  - **Day**—4
  - **Hour**—8
  - **Minute**—16
  - **Second**—32

**Note:** If you set a bit in the UNKNOWNDATEMAP value for a date component, the corresponding bit is not set in the VALIDDATEMAP value.

**Index**

Number, starting with 0, that specifies an index into an array of the selected items in a multiple-value control such as a list of check boxes. To use the Index attribute, either first return the COUNT of the control to determine how many items are selected, or find the selected items by looping through the list and creating an array of the selected items.

**ItemSetIndex**

Number that specifies an index into the items that make up an itemset. An itemset is a group of data items that repeat. For example, a concomitant medication CRF might contain an itemset to capture the same set of data items about each concomitant medication the patient is using. Valid ItemSetIndex values are 0 or greater:

- 0—Indicates that the control is not an itemset or, if the control is an itemset, references the current data item within the itemset. For example, if the user presses Submit after entering data in the third data item in the itemset, 0 sets the index to the third data item.
- Any number greater than 0—Explicitly indexes the specified item within the itemset.

**VisitIndex** Number that specifies an index into a repeating formset. A repeating formset is most often used for unscheduled visits in which the same data is recorded each time the visit occurs. Valid VisitIndex values are 0 or greater:

- 0—Indicates that the formset is not repeating or, if the formset is repeating, references the current formset.
- Any number greater than 0—Explicitly indexes the specified occurrence of the formset.

**Value** Value to set.

**Example**

This example uses SetValue with a special keyword, "EMPTY," to clear the value of a data item.

```
v2subno = Cint(Patient.Getvalue(Patient.GetCurPath(), "", 0, 0, 0))
```

```
Patient.SetEnteredValue v2subno, "", 0, 0, 0, EMPTY
```

**SetValueRF(REFNAMEPath,AttributeName,Index,ItemsetIndex,VisitIndex,FormIndex,Value)**

=-In instances of a repeating form, the SetValueRF method uses the normalized value of the data entered in the specified form component. The SetValueRF method is identical to the *SetValue* (see "*SetValue(REFNAMEPath,AttributeName,Index,ItemsetIndex,VisitIndex,Value)*" on page 312) method, except that it is intended for use in repeating forms. Therefore, it has an additional argument, FormIndex, which enables you to reference a specific instance of the repeating form.

**Note:** Use SetValueRF only in calculations, not in form rules.

## Arguments

REFNAMEPath String that identifies the form component to use. Each REFNAME in the string is the REFNAME specified in the .xml file that contains the definition of the form component. The REFNAMEPath must be fully qualified; only the patient component of the path defaults to the current patient. The REFNAMEPath argument is in the following format:  
(0.Visit.Form.Section.Itemset.Item[.control[.control...]])

- *0*—Current patient.
- *Visit*—REFNAME of the visit, as specified in the .xml file that contains the visit definition.
- *Form*—REFNAME of the form, as specified in the .xml file that contains the form definition.
- *Section*—REFNAME of the section, as specified in the .xml file that contains the section definition.
- *Itemset*—REFNAME of the itemset, as specified in the .xml file that contains the itemset definition. An itemset is a group of data items that repeat. For example, a concomitant medication CRF might contain an itemset to capture the same set of data items about each concomitant medication the patient is using.
- *Item*—REFNAME of the item, as specified in the .xml file that contains the item definition.
- *Control*—REFNAME of the control, as specified in the .xml file that contains the control definition. To access an component of a group control, refer to each parent control in which the child component is nested. For example, to address one of two text controls within a group control, enter the REFNAME of the group control followed by the REFNAME of the text control, and separate the names with periods, as follows: *GroupControlRefname.TextControlRefname*.

**AttributeName** String that specifies the type of information to use, specified in double quotes (""). Valid AttributeNames are:

- **VALUE**—Gets the value of the form component. This attribute is the default; you can indicate it with empty double quotes.
- **VALIDDATEMAP** — Number that specifies whether a date is complete. The script uses the VALIDDATEMAP attribute of the GetValue method on the Patient object. VALIDDATEMAP returns an integer that contains a bitmask indicating which elements of the DateTimeControl are complete:
  - **Year**—1
  - **Month**—2
  - **Day**—4
  - **Hour**—8
  - **Minute**—16
  - **Second**—32

- **UNKNOWNDATEMAP** — Number that specifies whether a date has components that are marked Unknown. The script uses the UNKNOWNDATEMAP attribute of the GetValue method on the Patient object. UNKNOWNDATEMAP returns an integer that contains a bitmask indicating which elements of the DateTimeControl are unknown:
  - **Year**—1
  - **Month**—2
  - **Day**—4
  - **Hour**—8
  - **Minute**—16
  - **Second**—32

**Note:** If you set a bit in the UNKNOWNDATEMAP value for a date component, the corresponding bit is not set in the VALIDDATEMAP value.

### Index

Number, starting with 0, that specifies an index into an array of the selected items in a multiple-value control such as a list of check boxes. To use the Index attribute, either first return the COUNT of the control to determine how many items are selected, or find the selected items by looping through the list and creating an array of the selected items.

### ItemSetIndex

Number that specifies an index into the items that make up an itemset. An itemset is a group of data items that repeat. For example, a concomitant medication CRF might contain an itemset to capture the same set of data items about each concomitant medication the patient is using. Valid ItemSetIndex values are 0 or greater:

- 0—Indicates that the control is not an itemset or, if the control is an itemset, references the current data item within the itemset. For example, if the user presses Submit after entering data in the third data item in the itemset, 0 sets the index to the third data item.
- Any number greater than 0—Explicitly indexes the specified item within the itemset.

**VisitIndex** Number that specifies an index into a repeating formset. A repeating formset is most often used for unscheduled visits in which the same data is recorded each time the visit occurs. Valid VisitIndex values are 0 or greater:

- 0—Indicates that the formset is not repeating or, if the formset is repeating, references the current formset.
- Any number greater than 0—Explicitly indexes the specified occurrence of the formset.

**FormIndex** Number that specifies an index into a repeating form.

- 0—References the current repeating form.
- 1—References a non-repeating form.
- Any number greater than 1—References the specified occurrence of the repeating form.

**Value** Value to set.

## Example

In this example the value of a control identified by the strSetBox2Path variable is set to "\*\*\*\*\*"

```
strSetBox2Value = Patient.GetValueRF(strSetBox2Path,"",0,0,0,0)
If strSetBox2Value <> "" then
Patient.SetValueRF strSetBox2Path,"",0,0,0,0,"*****"
End If
```

## ShowDynamicForm

The ShowDynamicForm method inserts a dynamic form into a visit based on the outcome of a calculation on another form.

ShowDynamicForm has the following parameters:

- RefName of the visit in which to schedule the appearance of the form
- RefName of the form to schedule
- Visit Index. This is a number that specifies an index into a repeating visit. Values are:
  - 0—Indicates that the visit is not repeating or, if the visit is repeating, references the current instance of the visit.
  - Any number greater than 0—Explicitly indexes the specified occurrence of the visit.

## Example

In this example, the patient's gender determines whether a pregnancy test form appears in the visit.

```
MALE=1
FEMALE=2

Gender=Patient.GetValue("0.visit1.DEM.DEM.0.GENDER", "", 0, 0, 0)
If Gender=FEMALE Then
Patient.ShowDynamicForm "visit1" , "Preg", 0
Else
Patient.RemoveNotStartedDynamicForm "visit1", "Preg", 0
End If
```

The rule is attached to the GENDER item on the DEM form as a Special Visit context.



## CHAPTER 4

# InForm Data Import utility

### In this chapter

Overview of the InForm Data Import utility.....	320
Importing a data and map file.....	323
Importing an XML file.....	341
Importing coded items.....	363
Date and time validation.....	368
Additional InForm Data Import utility attributes .....	369
Running the InForm Data Import utility from the command line.....	370
Enhancing your data import .....	372

# Overview of the InForm Data Import utility

## Parts of the InForm Data Import utility

Part	Description
Data file	<p>A pipe-delimited file that contains the following:</p> <ul style="list-style-type: none"> <li>• Line feed characters and carriage returns between lines.</li> <li>• Data to load into the trial database.</li> </ul>
Map file	A file that contains all the mapping information that is necessary to import information from the data file to the InForm database.
XML file	A file that contains information that can be directly imported into the trial database. For more information, see <i>Creating a data and map file</i> (on page 323) and <i>Adding new patient clinical data</i> (on page 346).
Rules	<p>Information against which the XML file is checked to ensure that it complies with the trial standards.</p> <p>You use rules to:</p> <ul style="list-style-type: none"> <li>• Avoid collisions during data entry.</li> <li>• Check the validity of data before it is committed to the trial database.</li> </ul> <p><b>Note:</b> Running rules might cause the import to run slowly.</p>
InForm application server	The server on which the InForm software is running.
Trial database	The database in which the data for the trial resides.



## Import methods

The following data loading methods are available in the InForm Data Import utility:

- **Importing a data file**—Uses a map file to define mappings between the imported data and the InForm database tables. The InForm Data Import utility then loads the data from a pipe-delimited ( | ) file into the InForm database.

When you use this method, the data is processed by the InForm application server, which runs edit checks to validate the data before writing to the database.

- **Importing an XML file**—Loads data from an XML file into the InForm database or transfers selected patient records from one site to another.

When you use this method, you can run rules during the import. If you run rules, the application processes the data as if you were entering it online; it runs edit checks and generates queries on data that fails the checks.

- **Importing an AutoCode XML file**—Similar to the MedML import option. Use this option for autocoding. For more information, see *Exporting coded controls* (on page 377).

**Note:** The InForm Data Import utility does not support importing RegDocs or Visit Reports data.

## Special considerations

- **Importing new patient data**—You must use the XML file option to import data for a patient who has not gone through the screening and enrollment process. After importing the data, you must stop and restart the trial to view the data online with the InForm application.

You cannot use the data and map file option to import screening and enrollment data. With the InForm Unplugged application, you can import screening and enrollment data only to a site server.

**Note:** You cannot import new data to a form with a **Frozen or Locked** status in the InForm application.

- **Importing comments**—To import form-level comments, you must use the XML file option. To import item-level comments, you must use either the data and map file option or the XML file option.
- **Importing calculated controls**—When you import calculated controls, the data type definition of the import field must be text control. If integer or floating number is used, the match is not recognized and data is not updated. Instead, the data is added as a new row.
- **Importing units**—When you submit unit data, the units must be associated with the previous field in the data and map file.
- **Importing unscheduled visit data**—To import data to unscheduled visits, use either the XML file or data and map file option.
- **Deleting and undeleting itemsets**—To delete or reinstate itemset data, you must use the XML file option.

- **Importing on multiple servers**—If your trial is running on multiple servers, run the InForm Data Import utility on only one server at a time to eliminate the possibility of importing duplicate data.
- **Editing repeating forms**—To create or edit repeating form instances, you must create the MAP file using a text editor, not the InForm Data Import utility user interface. Follow these guidelines for editing a map file:
  - Use NOFORMNEW in the first line of the MAP file to instruct the InForm Data Import utility not to create a new repeating form instance.
  - Use the new !formmatch! element in the MAP file to specify an item (not within an itemset) that will be compared to other repeating form instances.

You can use multiple !formmatch! elements. If all such elements match some existing repeating form instance, then that form instance will be updated. If there is no match (or no !formmatch! element) then a new repeating form instance will be created, as illustrated in the following sample code:

```
FORMNOVISITNEWNOFORMNEW |
!cd!Site |
!cd!Patient |
!visitmatch!0.UnschVisit.DOV.DOV.0.DOV.DOV!dtatetime! |
!formmatch!0.UnschVisit.HH.DH.0.DURATIONGROUP.DURTAIONGROUP.YRDUR
ATION!dtstring! |
!formmatch!0.UnschVisit.HH.DH.0.DURATIONGROUP.DURTAIONGROUP.MTHDU
RATION!dtstring! |
0.UnschVisit.HH.DH.0.previousgroup.previousgroup!DTSTRING! |
```

- When you create a repeating form, include regular items instead of itemsets to uniquely identify the form. If you cannot uniquely identify the form, then you can enter data only once to the form, and the next data entry will go to a new form instance.

# Importing a data and map file

## Creating a data and map file

The data and map file that used in the direct import method must be a text file. You can use any text editor that creates plain text files to create it, or develop a conversion tool that automatically formats your raw data.

You can direct the data in the file to target controls on more than one CRF; however, you must use separate files for each itemset into which you are importing rows of data, and data that you import into CRF itemsets must be in a separate file from data that you import into regular CRF items.

The import file must have the following characteristics.

- Each import row must include either of the following:
  - A field that specifies the patient number and initials in the following format:  
**patient\_number (patient\_initials)**  
 If it is possible that more than one patient could have the same patient number and initials, you must also include a field for the site mnemonic of the patient.
  - The database ID of the patient.
- All import fields must be separated by a pipe character ( | ).
- Do not include double quotation marks (") in the data file.

Additionally, some types of data must be presented in specific ways. The following sections describe how to set up the following fields and controls for import:

- *Date fields* (on page 323).
- *Time fields* (on page 324).
- *DateTime fields* (on page 324).
- *Nested Controls* (on page 324).
- *Checkboxes and multiple-selection drop-down lists* (on page 324).
- *Units* (on page 324).
- *Item comments* (on page 325).

## Date fields

Dates must consist of three fields in month|day|year format, using a 4-digit year. Observe these formatting considerations:

- If a date is missing a component, include a null field, as in the following example:

```
month | |year
```

- If a component of a date is unknown, use the keyword UNK, as in the following example:

```
month|UNK|year
```

## Time fields

Times must consist of three fields in hour|minute|second format, using a 24-hour clock. Observe these formatting considerations:

- If a time is missing a component, include a null field, as in the following example:

```
hour|minute|
```

- If a component of a time is unknown, use the keyword UNK, as in the following example:

```
hour|UNK|UNK
```

## DateTime fields

Dates and times must consist of six fields in month|day|year|hour|minute|second format, using a 4-digit year and a 24-hour clock. Observe these formatting considerations:

- If a datetime data item is missing a component, include a null field.
- If a component of a datetime item is unknown, use the keyword UNK.

Example:

```
PF|001 (CJB)|Sep|23|1975|1|1|215210|LABDATA|Clinical Chemistry:|AG
Ratio|XGR|Nov|UNK|1998|0.8-2|2.0|N||
PF|001 (CJB)|Sep|23|1975|1|1|215210|LABDATA|Clinical
Chemistry:|ALAT(SGPT)|XGP|Nov|UNK|1998|0-48|U/L|42|N||
PF|001 (CJB)|Sep|23|1975|1|1|215210|LABDATA|Clinical
Chemistry:|Albumin|XAL|Nov|UNK|1998|3.2-5|G/DL|4.3|N||
PF|001 (CJB)|Sep|23|1975|1|1|215210|LABDATA|Clinical Chemistry:|Alkaline
Phosphatase|XLK|Nov|UNK|1998|20-125|U/L|63|N||
```

## Nested Controls

Nested controls must consist of a field for each control separated by a pipe character ( | ). Only one of these fields will contain information for each row of data.

Example:

```
PF|001 (CJB)|Sep|23|1975|1|1|215210|LABDATA|hematology||
PF|001 (CJB)|Sep|23|1975|1|1|215210|LABDATA||urinalysis||
PF|001 (CJB)|Sep|23|1975|1|1|215210|LABDATA|||toxicology|
```

## Checkboxes and multiple-selection drop-down lists

Checkboxes and multiple-selection drop-down lists must consist of a field with a list of zero or more controls separated by a comma.

Example:

```
PF|001 (CJB)|Sep|23|1975|LABDATA|Sinus Tachycardia, Premature Ectopic Junctional
Beats, Sinus Bradycardia, Premature Ectopic Atrial Beats|
```

## Units

A field that contains units must immediately follow the value that it describes.

The example shows weight in pounds. The number value of the weight (177) is contained in one field, and the unit value (pounds) is contained in the following field.

```
PF|001 (CJB)|Sep|23|1975|70|in|177|lb|
```

## Item comments

Item comments must consist of a field that contains the text comment for the item.

Example:

PF|001 (CJB)|Sep|23|1975|70|177|weight measured with shoes and socks

## Specifying and editing map files

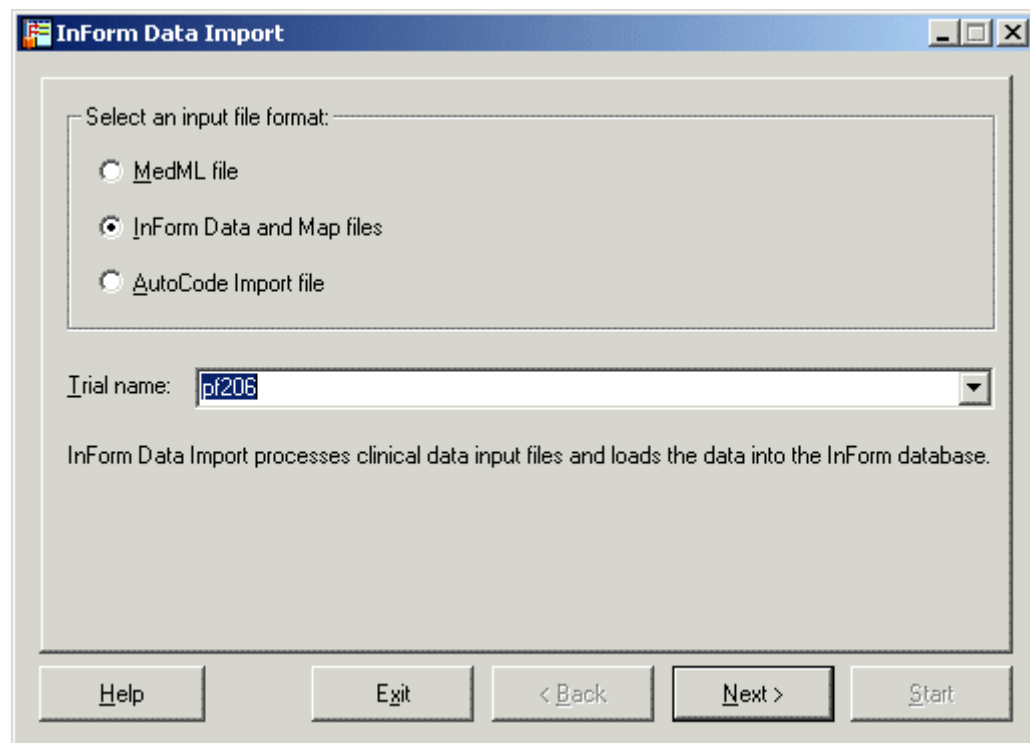
To import a data file, do one of the following:

- Specify an existing map file.
- Define a map file.

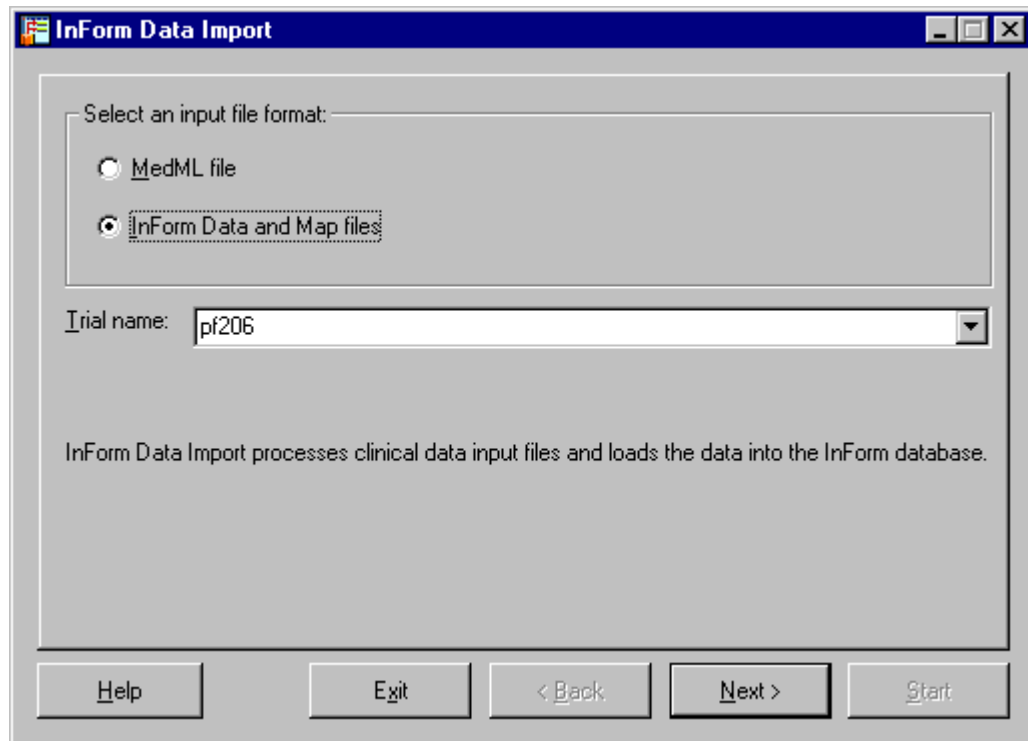
To specify and edit an existing import map file:

- 1 Select **Start > Programs > Oracle > InForm 4.6 > InForm Data Import utility**.

The InForm Data Import utility main window appears.



- 2 Select **InForm Data and Map files**.
- 3 In the **Trial Name** field, type or select the name of the trial into which to import the files.
- 4 Click **Next**.



- 5 Type the file name of the data file to import.
- 6 Type the file name of the map file to import.

**Note:** After you enter a file name and open the map file editor, the file is created, regardless of whether you click **Start** or **Stop**. This file is stored in the same location from which pfimport.exe was opened.

- 7 Click **Edit Map File**.

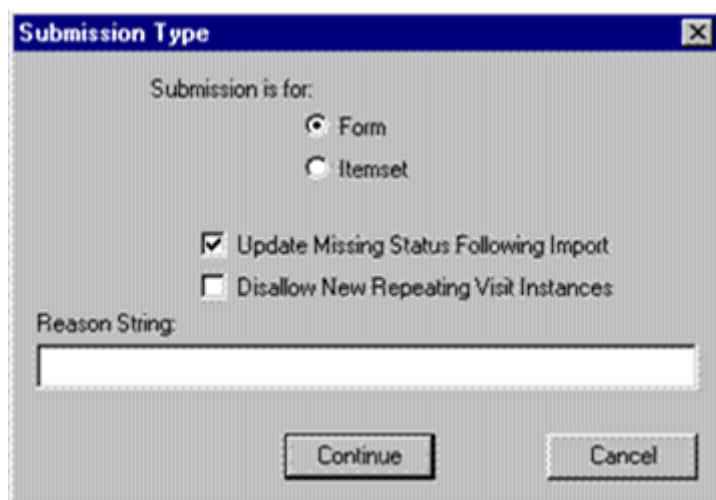
To create or edit a map file:

- Specify whether the data is targeted for CRF items or CRF itemsets. For more information, see *Specifying a submission type* (on page 327).
- Specify the input field type. For more information, see *Specifying an input field type* (on page 328).
- Specify the definition of each map file, one field at a time. For more information, see *Building an item path* (on page 329).
- Indicate whether the data contains multiple selection items. For more information, see *Indicating data contains multiple selection items* (on page 331).
- Specify the data type. For more information, see *Specifying a data type* (on page 332).
- Specify mappings between the values in your import file and the values defined for the target data fields in the InForm database. For more information, see *Mapping strings and child controls* (on page 332).
- Save the map file. For more information, see *Saving the map file* (on page 339).

- 8 After you have saved the map file, exit the map file editor to return to the InForm Data Import utility window to import the data and map file. For more information, see *Running the import using the data and map import file* (on page 336).

## Specifying a submission type

Use the Submission Type window to specify a submission type for the import file.



To specify a submission type:

- 1 Select one of the following options:
  - **Form**—If all of the data in the import file is targeted for regular CRF items.
  - **Itemset**—If all of the data in the import file is targeted for CRF itemsets. Optionally, to specify that you want to import only data that updates existing itemsets, select **Disallow New Itemset Rows**.
- 2 If you want to automatically update the traffic lights on a form when the import is complete and the trial is restarted, select **Update Missing Status Following Import**.
- 3 If you want to import data that only updates existing unscheduled visits, select **Disallow New Repeating Visit Instances**.
- 4 In the **Reason String** field, type the text that should appear in the **Reason for Change** section on the Data Value(s) screen if data is updated in the data load. The default reason is Lab Import.
- 5 Click **Continue**.

The Field Definition dialog box appears. Follow the procedure in *Specifying an input field type* (on page 328) to define map fields.

## Specifying an input field type

Use the Field Definition dialog to define each map field one at a time. Each map field corresponds to a data item in the import file.

The screenshot shows the 'Field Definition - Field 8 of 13' dialog box. It has a title bar with a close button. The main area contains a 'Sample Data' text box with '04/17/2000'. Below it is a radio button selected for 'InForm Item Path' with a text box containing 'D.Visit1.LAB.LAB.LAB.DATEOFTEST.STARTDATE' and a 'Build Path' button. To the right are three checkboxes: 'Comma separates multiple-values', 'Match Itemset instance with this field', and 'Match Repeating visit instance with this field'. Below these are five radio buttons: 'Patient Field - Number (Initials)', 'Site Mnemonic Field', 'Ignore This Field', 'Unit Symbol for previous field', and 'Comment Field for previous field'. A 'Map Strings' button is to the right of the last three radio buttons. At the bottom is a 'Data Type' section with buttons for 'String', 'Integer', 'Float', 'Date' (selected), 'Time', and 'DateTime'. At the very bottom are buttons for '< Back', 'Next >', 'Finish', 'Insert Field', 'Delete Field', and 'Cancel'.

In the Field Definition dialog box, select one of the following field types:

- **InForm Item Path**—Contains data that is targeted to a data item on a CRF. When you select this option, you must provide additional information about the import field. For more information, see *Building an item path* (on page 329).
- **Patient Field – Number (Initials)**—Contains patient identification information in either patient\_number (patient\_initials) format or in the form of a patient database ID. If your import data identifies each patient by the patient database ID, select the **Field Contains Known Patient ID** checkbox.

When you select this option, the definition of the map field is complete, and you can move to another map field definition or save the map file and exit the map file editor. For more information, see *Navigating the map file* (on page 335).

- **Site Mnemonic Field**—Contains the mnemonic of the site where the patient is enrolled. When you select this option, the definition of the map field is complete, and you can move to another map field definition or save the map file and exit the map file editor. For more information, see *Navigating the map file* (on page 335).
- **Ignore This Field**—Indicates that you do not want the field to be imported. When you select this option, the definition of the map field is complete, and you can move to another map field definition or save the map file and exit the map file editor. For more information, see *Navigating the map file* (on page 335).



- **Unit Symbol for previous field**—Contains the symbol for the units that apply to the previous field in the import and map files. The symbol of a unit is the text that identifies the unit on the CRF, as defined in the SYMBOL attribute of the UNIT definition in the appropriate XML file. When you select this option, the definition of the map field is complete, and you can move to another map field definition or save the map file and exit the map file editor. For more information, see *Navigating the map file* (on page 335).
- **Comment Field for previous field**—Contains the comment text that is associated with a form or itemset. When you select this option, the definition of the map field is complete, and you can move to another map field definition or save the map file and exit the map file editor. For more information, see *Navigating the map file* (on page 335).

## Building an item path

If you selected **InForm Item Path** as the item type, you must specify the RefName path for the target CRF data item for the field in the import file.

To build an item path, select each RefName from the drop-down lists in the Build Path from Database dialog, or type the path in the **InForm Item Path** text box.

### Using the Build Path from Database dialog box

- 1 In the Field Definition dialog box, select **InForm Build Path**.
- 2 Click **Build Path**.

The Build Path From Database dialog appears.

- 3 From the **Visit** drop-down list, select the RefName of the target visit.
- 4 From the **Form** drop-down list, select the RefName of the target CRF.
- 5 From the **Section** drop-down list, select the RefName of the target section.
- 6 Optionally, from the **Itemset** drop-down list, select the itemset RefName to load the import data into an itemset.
- 7 From the **Item** drop-down list, select the RefName of the target item.

**Note:** You must create a separate map field definition for each item in an itemset.

- 8 From the **Control** and **Child Control** drop-down lists, select the RefName of the target group and child controls.
- 9 Click **OK**.

## Entering an item path explicitly

To specify an item path, use the following item path:

```
0.Visit.Form.Section.Itemset.Item[.control[.control...]]
```

Each component of the item path is the RefName used to define an element:

- **0**—Indicates the current patient.
- **Visit**—RefName of the visit, as specified in the XML file that contains the visit definition.
- **Form**—RefName of the CRF or other form, as specified in the XML file that contains the form definition.
- **Section**—RefName of the section, as specified in the XML file that contains the section definition.
- **Itemset**—RefName of the itemset, as specified in the XML file that contains the itemset definition. If the import data for which you are creating a map field definition is a regular CRF item, not an itemset, type 0.
- **Item**—RefName of the item, as specified in the XML file that contains the item definition. Create a separate map field for each item in an itemset.
- **Control**—RefName of the control, as specified in the XML file that contains the control definition. To access an element of a group control, refer to each parent control in which the child element is nested. For example, to address one of two text controls within a group control, type the RefName of the group control followed by the RefName of the text control, and separate the names with periods, as follows: GroupControlRefName.TextControlRefName.

Examples

- This example shows the item path for the TEMPTEXT field in the TEMPTEXT item in the VS section of the VSL form in a visit called VISIT1. Note the substitution of 0 for the Itemset RefName.

```
0.Visit1.VSL.VS.0.TEMPTEXT.TEMPTEXT
```

- This example shows the item path for an item in an itemset. The control is a date control called ONSETDATE, in the ONSETDATE item of the SS2GROUP itemset in the SECTION2 section of the SS form in VISIT1.

```
0.Visit1.SS.SECTION2.SS2GROUP.ONSETDATE.ONSETDATE
```

- This example shows the item path for a text control nested within the VIEW radio control item in the CHESTXRAY section of the ECG form in VISIT1. The item is identified by the VIEW RefName; the radio control is identified by VIEWRADIO, and the text control is identified by OTHERTEXT.

```
0.Visit1.ECG.CHESTXRAY.0.VIEW.VIEWRADIO.OTHERTEXT
```

## Indicating that data contains multiple selection items

To import data to a multiple-selection control, a checkbox group, or a multiple-selection drop-down list:

- 1 In the import file, include all applicable selections in a single field. Separate each selection with a pipe (|).

For example, if you want the cigarettes and cigars checkboxes to be selected in a list containing cigarettes, cigars, and Not Done, and the values defined for those selections are “cigarettes,” “cigars,” and “ND,” the import file should contain a field with the following value:

```
|cigarettes,cigars|
```

- 2 Define a map field as an InForm Item Path field that references the parent control for the checkboxes or the drop-down list.
- 3 In the Field Definition dialog, select the **Comma Separates Multiple Values** checkbox.

## Checking for duplicate information within itemsets

The InForm Data Import utility can determine whether data already exists in the database by comparing the data itemset in which you are mapping to existing itemsets in the database.

To use this feature, in the Field Definition dialog, select **Match Itemset instance with this field** for each control.

For example, if you are importing lab information and the patient name, the data, and the type of test match data are already in the database, this might indicate that the data is a duplicate. The InForm Data Import utility recognizes this as duplicate data and does not add a second instance of the data in the database.

- 1 In the Submission Type window, select **Itemset**, then click **Continue**.  
The Field Definition window appears.
- 2 Select the **Item Path** for the data object you want to match.
- 3 Select **Match Itemset instance with this field**.
- 4 Repeat these steps for any other data items you want to compare.

## Importing information into an unscheduled visit

To import information into an unscheduled visit:

- 1 In the Submission Type window, select **Itemset**.
- 2 Click **Continue**.  
The Field Definition window appears.
- 3 Select the **Item Path** for the date of visit with which you want to match a repeating visit.
- 4 Select **Match Repeating visit instance with this field**.

Repeat these steps for any other unscheduled visits to import.

## Specifying a data type

You must specify a data type for each import field definition that you create as an InForm Item Path field.

To specify the data type of an import field:

- Click the appropriate button in the **Data Type** group.

The InForm Data Import utility issues an error if any of the following exist:

- An invalid integer field. An invalid integer field cannot be fully converted to an integer value. For example, 123\$ is an invalid integer field.
- A string identified as a floating number that does not meet the specification for the CRF control for allowed number of digits before and after the decimal point.
- A string identified as a text control that is not within the defined size range for the CRF text control.

## Mapping strings and child controls

Use the InForm Data Import utility to map field values in your import file to the database definitions of CRF controls, which have predefined values. Additionally, the mapping feature generates mappings in compound controls between individual child controls and their database ID paths.

### Mapping strings

When the target of an input data field is a control for which an online user selects a predefined value, the value of the import field must be the same as the value of the selected control as defined in the database. These values are case-sensitive.

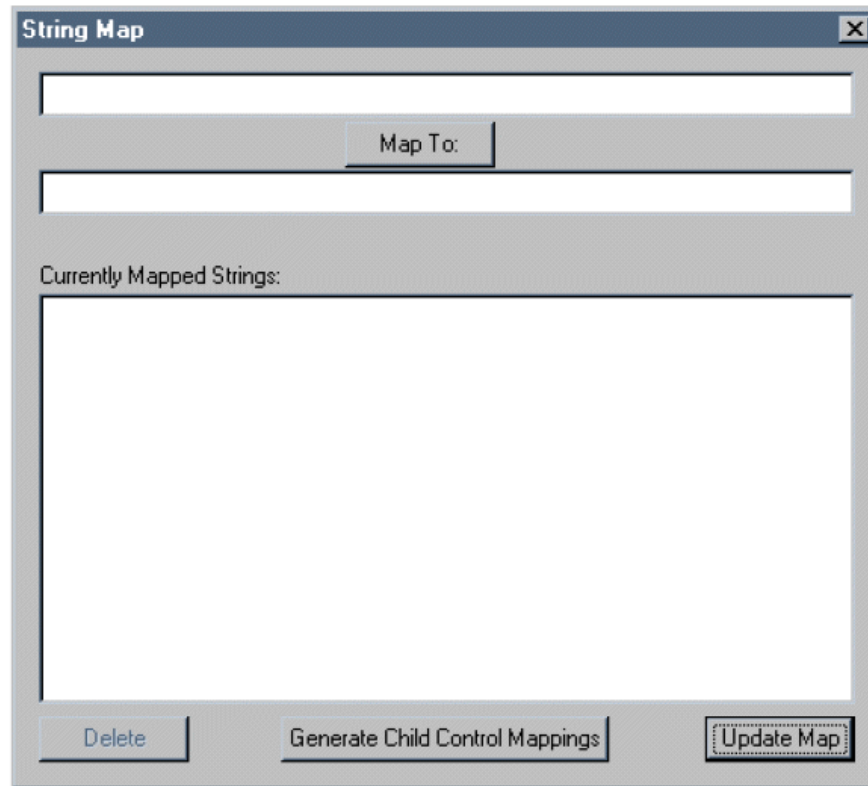
If your import file does not match the defined database values, you can convert your file to match them.

Alternatively, you can use the string mapping feature of the InForm Data Import utility to specify mappings between the values in your import file and the values defined for the target data fields in the InForm database.

To use this feature while creating the definition of a map file field:

- 1 In the Field Definition dialog box, click **Map Strings**.

The String Map dialog box appears.



- 2 In the top field, type a possible value of the control as it appears in the import file.
- 3 In the next field, type the value of the control as it is defined in the database. This definition is specified, generally with a VALUE attribute, in the XML file that is used to load form and data item definitions into the database.
- 4 Click **Map To**. The utility transfers the pair of values to the **Currently Mapped Strings** field. For example, if you typed *Yes* as a value that appears in your file and *Y* as the defined control value, the **Currently Mapped Strings** field shows the mapping as *Yes maps to Y*.
- 5 Repeat the mapping definition for each possible combination of values that the field can have in your import file and in the database definition of the control.
- 6 Click **Update Map**.

## Mapping child controls

When the target control is nested within another control (for example, a field within a list of radio buttons), you must create separate map fields for the group control and for each child control within the group. Similarly, your import file must contain fields for the group control and for each possible child control selection.

To assign a specific value to the group control selection, the InForm Data Import utility maps child control names to their database ID paths.

To generate child control mappings for a group control map field:

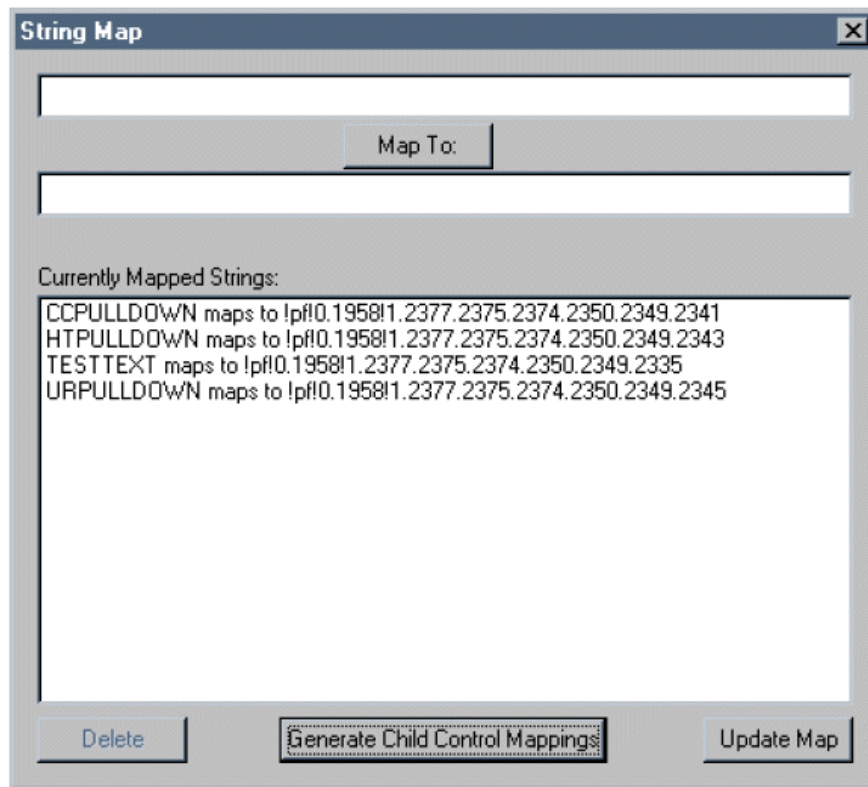
- 1 In the Field Definition dialog box, click **Map Strings**.

The String Map dialog box appears.

- 2 Click **Generate Child Control Mappings**.

The **Currently Mapped Strings** field shows the mappings between child control RefNames and their database IDs. In the import file field that corresponds to the map field that defines the group control, type the child control RefName for which you are providing data.

- 3 Click **Update Map**.



## Navigating the map file

As you create the fields in a map file, the field definitions are strung together in a sequence in which you can move back and forth.

Use the control buttons at the bottom of the Field Definition dialog box to do the following:

- To view a field that occurs earlier in the map file, click **Back**.
- To view a field that occurs later in the map file, click **Next**.
- To create a new field definition, advance to the last field in the map file and click **Create Next**.

**Note:** You cannot navigate away from the field until you create or delete the new field definition by clicking **Finish** or **Delete**.

## Inserting or deleting an import field

To insert or delete an import field definition in the map:

- 1 In the InForm Data Import utility dialog, in the **Field Map File** field, type the name of the file.
- 2 Click **Edit Map File**.
- 3 Click **Insert Field**.

**Note:** The InForm Data Import utility inserts the new field definition immediately before the field that is displayed.

The InForm Data Import utility clears the data entry fields on the Field Definition dialog.

- 4 Type the definition for the new field.
- 5 Click **Next**, **Back**, or **Finish**, as appropriate.

The InForm Data Import utility inserts the new field definition immediately before the field that was displayed when you clicked **Insert**.

- 6 To save the map definition, click **Finish**.

**Note:** You cannot navigate away from the field until you create or delete the new field definition by clicking **Finish** or **Delete Field**.

To delete an import field definition:

- 1 Click **Next** or **Back** to find the field definition to delete.
- 2 Click **Delete Field**.
- 3 To save the map definition, click **Finish**.

**Note:** If you change the definitions in a map file and click **Cancel** (instead of **Finish**), the InForm Data Import utility saves an empty map file with the path and filename that you specified in the **Map** field.

## Checking the map against the import file

Use the InForm Data Import utility map file editor to review the map field definitions against the actual import data that you will be processing.

To check the map file against the first line of the import file:

- 1 In the InForm Data Import utility dialog, in the **Map** field, type the name of the map file to check, or click **Browse** and locate the file.
- 2 In the **Data** field, type the name of the import file, or click **Browse**.
- 3 Click **Edit Map File**.
- 4 In the Submission Type window, click **Continue**.
- 5 In the Field Definition dialog, navigate through the field definitions and compare the definition of each file with the data that appears in the **Sample Data** field.

## Running the import using the data and map import file

When your map definition is complete and the import file and map file are synchronized, you can import data.

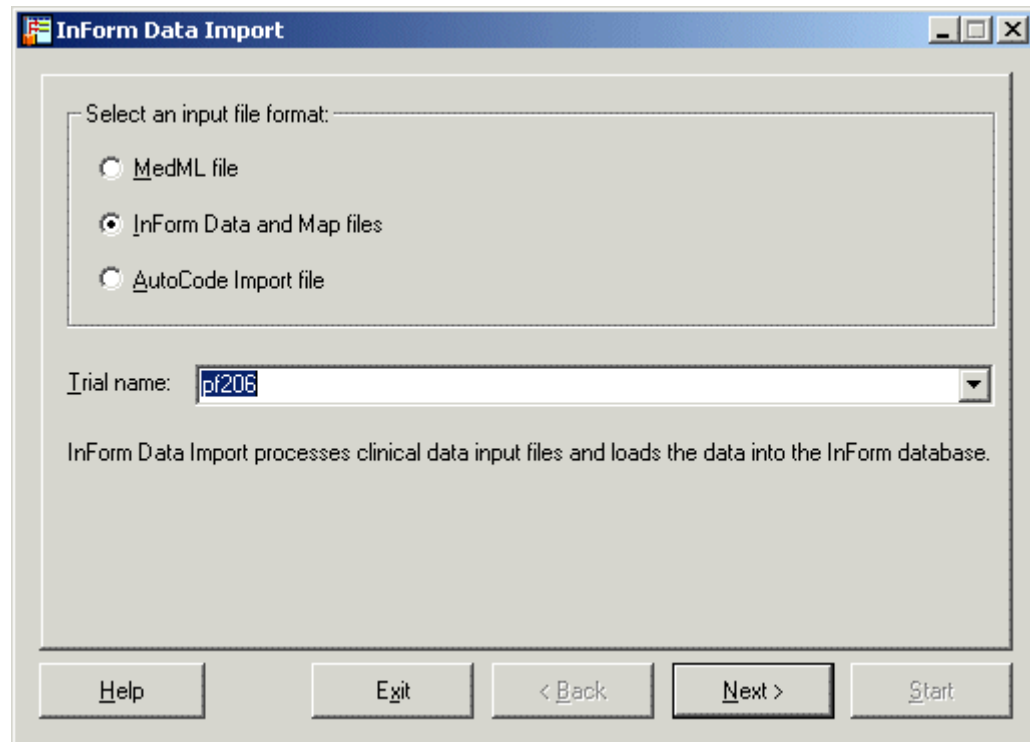
**Note:** To import data into the database, the server must be running before you start the InForm Data Import utility. To see the effect of imported data on the patient status icons, you must stop and restart the server after importing.

To import InForm data and map files into the InForm database:

- 1 Double-click the **PFIimport.exe** file located in the **\bin** directory of the InForm installation.



The InForm Data Import utility appears.



- 2 Select **InForm Data and Map files**.
- 3 In the **Trial Name** field, select or type the name of the trial into which you want to import the InForm data and map files.
- 4 Click **Next**.
- 5 In the **Data** field, type the full path name of the data file you want to import, or click **Browse**.
- 6 In the **Map** field, type the full path name of the map file you want to import, or click **Browse**.
- 7 To create or edit the map file, click **Edit Map File**. For more information, see *Specifying and editing map files* (on page 325).
- 8 Select **Next**.

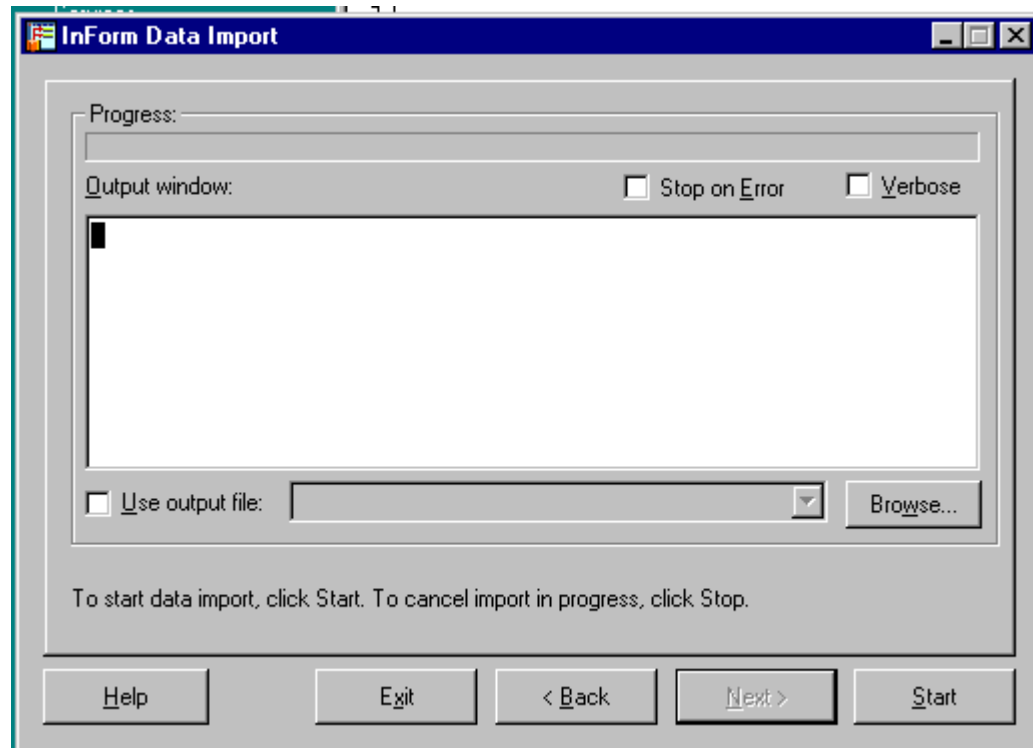
A dialog appears and requests your InForm name and password.

- 9 In the **Name** field, type the name of an InForm user who has the appropriate rights for the data you are importing:

To import data that matches this InForm system activity	User needs these rights
Add patient clinical data	Enter Data into a CRF
Update patient clinical data	Edit Data on a CRF

- 10 In the **Password** field, type the user password, then click **Next**.

The Summary window appears.



- 11 Optionally, select any of the following:
  - **Stop on Error**—To instruct the InForm Data Import utility to stop if it encounters an error.
  - **Verbose**—To instruct the InForm Data Import utility to generate detailed messages as it processes the file.
  - **Use output file**—Specify the filename to save the output file as a text file.
- 12 Click **Start**.

The InForm Data Import utility processes the import file, writes messages to the message area and the output file, if specified, and adds or updates data in the database.

- 13 Close the InForm Data Import utility. To view the data that you imported, you must stop and restart the trial.

## Saving the map file

To save the map definitions in the specified file:

- In the **Field Map File** field, click **Finish**.

**Note:** If you change definitions in a map file and click **Cancel** (instead of **Finish**), an empty map file is saved with the path and filename you specified in the Map field.

## Checking the error file

As the InForm Data Import utility processes the import file, it creates an error file if it is unable to import any of the import file rows.

To check the error file:

- Open the output file with the filename that you specified, in the directory that you specified.

**Note:** If you did not specify a directory and filename for the output file, the error file is saved in the same directory in which the InForm Data Import utility executable, PFIImport.exe, is stored. After the import is complete, check for the presence of an ERR file and review the file for errors.

## Importing an XML file

To import an XML file:

- 1 Create an XML file that contains the patient data to add or update. For more information, see *Creating an XML file for data import* (on page 341).
- 2 Run the import with the MedML file option. For more information, see *Running the import using the XML import file* (on page 359).

### XML import files

The import file for the MedML file option is an XML file that contains elements that specify the type of processing to perform during the import and the destinations and values of the import data. The file can contain tags for the following types of import actions:

- Screening and enrolling a patient.
- Adding new patient data.
- Updating existing patient data.
- Transferring a patient from one site to another.

To create the import file, use any text editor that creates plain text files.

### Creating an XML file for data import

To create an XML file for data imports:

- 1 Create a first line that contains the XML version number. The version string must be lower case:  

```
<?xml version="1.0"?>
```
- 2 Add an opening and closing element that tells the InForm Data Import utility what type of processing to perform:  

```
<CLINICALDATA>
</CLINICALDATA>
```
- 3 Between the opening and closing elements, add opening and closing elements for each activity for the InForm Data Import utility to perform. Use one set of activity elements for each patient for whom to import data. For example, to import screening data for a new patient, use the following elements:  

```
<SCREEN>
</SCREEN>
```

- 4 In the opening element for the import activity, add the attributes required for that activity type, and any optional attributes. For example, the SCREEN element requires a SITEMNEMONIC or SITENAME attribute to specify the patient site by mnemonic or by name. If the patient site mnemonic is PF, you would insert the SITEMNEMONIC element as follows:

```
<SCREEN SITEMNEMONIC="PF" >
```

**Note: If you are using the InForm Data Import utility to transfer patient records between sites, go to the final step. A patient record transfer import file does not use the DATA element.**

- 5 Between the opening and closing elements that specify the import activity, insert a DATA element for each form control:

```
<DATA />
```

- 6 The DATA tag has the following required attributes:

- **TAG**—A database path that identifies the target data item control, and that is made up of RefNames in the following order:  
Section.Itemset.Item[.control[.control...]]
- **Section**—RefName of the section, as specified in the XML file that contains the section definition.
- **Itemset**—RefName of the itemset, as specified in the XML file that contains the itemset definition. If the target data item is a regular CRF item, not an itemset, type 0.
- **Item**—RefName of the item, as specified in the XML file that contains the item definition. Create a separate DATA element for each item in an itemset.
- **Control**—RefName of the control, as specified in the XML file that contains the control definition. To access an element of a group control, refer to each parent control in which the child element is nested. For example, to address one of two text controls within a group control, type the RefName of the group control followed by the RefName of the text control, and separate the names with periods, as follows:  
GroupControlRefName.TextControlRefName.
- One of the following:
  - **VALUE**—The value of the data to import into the control. Enclose the value in double quotes.

**Note: Because double quotes are used to delimit the value of an attribute, you can not include double quotes as part of the value text. If you need to include double quotes as part of the value text, use the XML entity reference &quot;.**

- **CHILDSELECTED**—The RefName of the selected child control, if the child control is nested within a compound control. For example, use the CHILDSELECTED attribute to indicate which radio control to select if the radio control includes two drop-down lists.
- **MONTH, DAY, YEAR, HOUR, MINUTE, SECOND**—The value of each applicable part of a datetime control.
- **UNIT**—Unit type of the selected control, when a unit definition is part of the target control.

- **COMMENT**—The text of an item-level comment.
- **REASONINCOMPLETE**—The reason the item is incomplete. When you specify this attribute, do not include a VALUE or any datetime control attributes in the DATA tag.
- **NOMULTIVALUE**—Indicates that a data value containing a comma (,) is a single value. Without this attribute, only the data preceding a comma is stored as a value. Data that occurs after a comma is assumed to be a separate value in a multi-value DATA tag.

For existing patient data only:

- **CLEARVALUE**—Clears the existing value for the specified control. Available options are TRUE or FALSE.

7 Save the file.

## Examples

The following example shows a DATA element that is used to import the initials of patient AAA in the PF site to the patientinitials field in the screening form:

```
<SCREEN SITEMNEMONIC="PF">
 <DATA TAG="screen.0.patientinitials.patientinitials"
 VALUE="AAA" />
</SCREEN>
```

The following example shows the use of the CHILDSELECTED attribute of the DATA element to indicate that, in the RACE radio control, the RACEPULLDOWN radio button is being selected. The second DATA element gives the selected value within the drop-down list.

```
<PATIENTDATA PATIENTINITIALS="AAA" SITEMNEMONIC="PF"
FORMSETREFNAME="Visit1" FORMREFNAME="DEM" COMMENT>
 <DATA TAG="DEM.0.RACE.RACEGROUP"
 CHILDSELECTED="RACEPULLDOWN" />
 <DATA TAG="DEM.0.RACE.RACEGROUP.RACEPULLDOWN"
 VALUE="Asian" />
</PATIENTDATA>
```

The following example shows a DATA element used to specify a date to be imported into an enrollment form:

```
<ENROLL PATIENTINITIALS="AAA" SITEMNEMONIC="PF"
PATIENTNUMBER="BK1" ENROLL="TRUE">
 <DATA TAG="consent.0.consentdate.date"
 MONTH="1" DAY="6" YEAR="1999" />
</ENROLL>
```

The following example shows the use of the UNIT attribute to specify that the unit in which height is being measured is inches.

```
<PATIENTDATA PATIENTINITIALS="AAA" SITEMNEMONIC="PF"
FORMSETREFNAME="Visit1" FORMREFNAME="DEM">
 <DATA TAG="DEM.0.HEIGHT.HEIGHTTEXT"
 VALUE="67" UNIT="Inches" />
</PATIENTDATA>
```

The following example shows the use of the NOMULTIVALUE attribute to specify that a data value containing a comma (,) is a single value.

```
<PATIENTDATA PATIENTINITIALS="AAA" SITEMNEMONIC="PF"
FORMSETREFNAME="Visit1" FORMREFNAME="DEM">
 <DATA TAG="DEM.0.HEIGHT.ID"
 VALUE="67,11" NOMULTIVALUE="" />
</PATIENTDATA>
```

## Screening

To import screening data for a patient at a site, use the SCREEN element with the following required attribute:

Attribute	Definition
SITEMNEMONIC	Specifies the mnemonic of the site at which the patient is being screened. Either SITEMNEMONIC or SITENAME is required.

### Example

The following sample file illustrates the elements used to screen patient AAA at site PF.

```
<?xml version="1.0"?>
<CLINICALDATA>
<!-- Screen Patient -->
<SCREEN SITEMNEMONIC="PF">
 <DATA TAG="screen.0.patientinitials.patientinitials"
 VALUE="AAA" />
 <DATA TAG="screen.0.eligible.eligible" VALUE="yes" />
 <DATA TAG="screen.0.datescreened.date" MONTH="1" DAY="6"
 YEAR="1999" />
 <DATA TAG="screen.0.dob.dob" MONTH="11" DAY="11"
 YEAR="1959" />
</SCREEN>
</CLINICALDATA>
```

## Enrolling

To import enrollment data for a patient at a site, use the ENROLL element with the following required attributes:

Attribute	Definition
PATIENTINITIALS	Specifies the initials of the patient that is being enrolled.
SITEMNEMONIC	Specifies the mnemonic of the site at which the patient is being screened. Either SITEMNEMONIC or SITENAME is required.
DUPLICATEORDER	Number that specifies the order in which patients who have the same patient initials and were enrolled in the same site were screened.
PATIENTNUMBER	Specifies the patient number of the patient being enrolled.
ENROLL	TRUE or FALSE, indicating whether to enroll the patient.



## Example 1

The following example illustrates the elements used to enroll patient XYZ at site PF. This example assumes that patient XYZ has previously been screened.

```
<?xml version="1.0"?>
<CLINICALDATA>

<!-- Screen Patient -->
<SCREEN SITEMNEMONIC="PF">
<DATA TAG="screen.0.patientinitials.patientinitials" VALUE="XYZ"/>
<DATA TAG="screen.0.eligible.eligible" VALUE="yes"/>
<DATA TAG="screen.0.datescreened.date" MONTH="1" DAY="6" YEAR="1999"/>
<DATA TAG="screen.0.dob.dob" MONTH="11" DAY="11" YEAR="1959"/>
</SCREEN>

<!-- Enroll Patient -->
<ENROLL PATIENTINITIALS="XYZ" SITEMNEMONIC="PF" PATIENTNUMBER="BK-XYZ"
ENROLL="TRUE">
<DATA TAG="consent.0.consentdate.date" MONTH="1" DAY="6" YEAR="1999"/>
<DATA TAG="consent.0.patientnumber.patientnumber" VALUE="28"/>
<DATA TAG="inclusion.0.age_inc.yesno" VALUE="1"/>
<DATA TAG="inclusion.0.hyper_inc.yesno" VALUE="1"/>
<DATA TAG="inclusion.0.understand_inc.yesno" VALUE="1"/>
<DATA TAG="inclusion.0.agree_inc.yesno" VALUE="1"/>
<DATA TAG="exclusion.0.secondary_ex.yesno" VALUE="0"/>
<DATA TAG="exclusion.0.malignant_ex.yesno" VALUE="0"/>
<DATA TAG="exclusion.0.allergyhistory_ex.yesno" VALUE="0"/>
<DATA TAG="exclusion.0.myocardial_ex.yesno" VALUE="0"/>
<DATA TAG="exclusion.0.monitor_ex.yesno" VALUE="0"/>
</ENROLL>
</CLINICALDATA>
```

## Example 2

The following example illustrates the use of the DUPLICATEORDER attribute to specify the order in which patients with duplicate patient initials should be enrolled. Patients John R. Doe and Jane R. Doe (JRD) are screened at site PF; first John, then Jane.

```
<!-- This is John R. Doe-->
<ENROLL PATIENTINITIALS="JRD" SITEMNEMONIC="PF" DUPLICATEORDER="1"
PATIENTNUMBER="BK-JRD" ENROLL="TRUE">
</ENROLL>
<!-- This is Jane R. Doe-->
<ENROLL PATIENTINITIALS="JRD" SITEMNEMONIC="PF" DUPLICATEORDER="2"
PATIENTNUMBER="BK-JRD" ENROLL="TRUE">
</ENROLL>
```

## Adding new patient clinical data

To add new clinical data to an existing patient at a site, use the PATIENTDATA element with the following attributes:

Attribute	Definition
PATIENTINITIALS	Specifies the initials of the patient that is being enrolled. Either PATIENTINITIALS or PATIENTNUMBER is required.
PATIENTNUMBER	Specifies the patient number of the patient being enrolled. Either PATIENTINITIALS or PATIENTNUMBER is required.
SITEMNEMONIC	Specifies the mnemonic of the site at which the patient is being screened. Either SITEMNEMONIC or SITENAME is required. Either SITEMNEMONIC or SITENAME is required.
SITENAME	Specifies the name of the site at which the patient is enrolled. Either SITEMNEMONIC or SITENAME is required.
FORMSETREFNAME	Specifies the RefName of the visit to which you are importing data.
FORMSETINDEX	Indicates to which visit instance to add the data. Either FORMSETINDEX or FORMSETINDEXORDER is required.
FORMSETINDEXORDER	Indicates to which visit instance to add the data. Either FORMSETINDEX or FORMSETINDEXORDER is required.
FORMREFNAME	Specifies the RefName of the CRF to which you are importing data.
REASONINCOMPLETE	This attribute may apply to either the form or item level. The value is one of the values in the radio group control. This attribute will be ignored if the form or item is complete.  To add an incompleteness reason at the form level, do not include a DATA tag in the PATIENTDATA group.
FORMINDEX	(Optional) If not present, then a new repeating form instance will be created (if FORMREFNAME is a repeating form). If present, the value indicates to which form instance the new data will be added.
ASSOCIATION	Creates or deletes an association instance.

## Adding data to an itemset

To add data to an itemset on a form, use the following additional attributes:

Attribute	Definition
SECTIONNAME	Specifies the RefName of the section in which the itemset occurs. Either SECTIONNAME or SECTIONID is required.
SECTIONID	Specifies the ID of the section in which the itemset occurs. Either SECTIONNAME or SECTIONID is required.
ITEMSETNAME	Specifies the RefName of the itemset definition.
ITEMSETINDEX	Number of the itemset line to update. If the ITEMSETINDEX section is blank or has a value of 0, the InForm Data Import utility adds a new itemset line. If ITEMSETINDEX is a number other than 0, the InForm Data Import utility updates the specified line. Note that existing itemset data is not changed, but missing data is filled in. For more information, see <i>Example 2: Creating an itemset row</i> (on page 349).

## Adding data to an unscheduled visit

To add data to an unscheduled visit, use the following additional attributes:

Attribute	Definition
NEWUNSCHEDVISIT	Indicates whether the data is being added to a new unscheduled visit. Values are TRUE or FALSE (the default). Use this attribute on the Visit Date form, a predefined form with the FORMREFNAME of DOV. For more information, see <i>Example 3: Adding data to an unscheduled visit</i> (on page 349).
FORMSETINDEX	Number that specifies the unscheduled visit to which to add the data. The number corresponds to the order in which the unscheduled visit was added to the trial.

## Importing autocoded data

When you use the InForm Data Import utility to import data that was exported using the autocode feature of the InForm application and coded by an external coding application, the InForm Data Import utility sets the following attribute to TRUE:

Attribute	Definition
AUTOCODEIMPORT	<p>Indicates whether data has been exported using the autocode feature of the InForm application and is being imported after being coded by an external coding application.</p> <p>A value of TRUE indicates that the InForm application must determine whether signatures on the form that is receiving the data must be invalidated. This determination is based on:</p> <ul style="list-style-type: none"> <li>• Visibility of the coded data.</li> <li>• Setting of the INVALIDATIONLEVEL attribute of the SIGNCRF MedML tag that associates a form with a signature group.</li> </ul> <p>For more information, see the <i>Setting Up a Trial with InForm Architect and MedML Guide</i>.</p>

## Adding a comment

You can add a comment to an item, an item within an itemset, or a form by using the COMMENT attribute:

- To add a comment to an item, include the COMMENT attribute along with the text of the comment in the DATA element.
- To add a comment to a form, include the COMMENT attribute along with the text of the comment in a PATIENTDATA element that does not include SECTIONNAME and ITEMSETNAME attributes.

### Example 1: Adding data to a form

The following XML fragment shows elements used to add data to the DEM form for patient AAA at site PF.

```
<?xml version="1.0"?>
<CLINICALDATA>
<PATIENTDATA
 PATIENTINITIALS="VOL" SITEMNEMONIC="PF"
 FORMSETREFNAME="Visit1"
 FORMREFNAME="DEM" COMMENT="This form was edited by Joe">
 <DATA TAG="DEM.0.GENDER.GENDERRADIO" VALUE="1"/>
 <DATA TAG="DEM.0.DEMDOB.dob" MONTH="2" DAY="14"
 YEAR="1961" COMMENT="As reported by patient"/>
 <DATA TAG="DEM.0.RACE.RACEGROUP"
 CHILDSELECTED="RACETEXT"/>
 <DATA TAG="DEM.0.RACE.RACEGROUP.RACETEXT"
 VALUE="African American"/>
 <DATA TAG="DEM.0.HEIGHT.HEIGHTTEXT" VALUE="67"
 UNIT="Inches"/>
 <DATA TAG="DEM.0.WEIGHT.WEIGHTTEXT" VALUE="150"
 UNIT="Pound"/>
 <DATA TAG="DEM.0.FRAME.FRAMEPULLDOWN" VALUE="1"/>
```

```

<DATA TAG="SH.0.SMOKE.SMOKERADIO" VALUE="Y" />
<DATA TAG="SH.0.EVERSMOKED.SMOKERADIO" VALUE="N" />
<DATA TAG="SH.0.WHATSMOKED.SMOKECHECKBOX"
 VALUE="cigarette,pipe" />
<DATA TAG="SH.0.HOWMUCHSMOKED.SMOKERADIO2"
 CHILDSELECTED="NUMTEXT" />
<DATA TAG="SH.0.HOWMUCHSMOKED.SMOKERADIO2.NUMTEXT"
 VALUE="10" />
<DATA TAG="SH.0.YRSSMOKED.SMOKERADIO2"
 VALUE="NDElement" />
</PATIENTDATA>
</CLINICALDATA>

```

## Example 2: Creating an itemset row

The following XML fragment shows the elements used to create a new itemset row on the Hypertension History form.

```

<PATIENTDATA
 PATIENTINITIALS="AAA" SITEMNEMONIC="PF"
 FORMSETREFNAME="Visit1" FORMREFNAME="HH"
 SECTIONNAME="PT" ITEMSETNAME="PT">
 <DATA TAG="PT.PT.THERAPYTEXT.THERAPYTEXT"
 VALUE="aspirin" />
 <DATA TAG="PT.PT.DOSAGETEXT.DOSAGETEXT"
 VALUE="1 tablet" />
</PATIENTDATA>

```

The following XML fragment shows the elements used to finish entering items for an existing itemset on the Hypertension History form.

```

<PATIENTDATA
 PATIENTINITIALS="AAA" SITEMNEMONIC="PF"
 FORMSETREFNAME="Visit1" FORMREFNAME="HH"
 SECTIONNAME="PT" ITEMSETNAME="PT" ITEMSETINDEX="1">
 <DATA TAG="PT.PT.DOSEDATE.DOSEDATE" MONTH="12" DAY="23"
 YEAR="1998" />
 <DATA TAG="PT.PT.DISCULLDOWN.DISCULLDOWN"
 VALUE="Not Effective" />
</PATIENTDATA>

```

## Example 3: Adding data to an unscheduled visit

The following XML fragment shows the elements used to add data to the DOV form and Vital Signs form in the first and second unscheduled visits containing those forms.

```

<?xml version="1.0"?>
<CLINICALDATA>
<!-- DOV form -->
<PATIENTDATA
 PATIENTINITIALS="ABC" SITEMNEMONIC="PF"
 FORMSETREFNAME="UnschVisit" FORMREFNAME="DOV"
 NEWUNSCHEDVISIT="TRUE">
 <DATA TAG="DOV.0.DOV.DOV" MONTH="2" DAY="1"
 YEAR="1999" />
</PATIENTDATA>
<!-- VitalSigns form -->
<PATIENTDATA
 PATIENTINITIALS="ABC" SITEMNEMONIC="PF" FORMSETINDEX="1"
 FORMSETREFNAME="UnschVisit" FORMREFNAME="VS">
 <DATA TAG="VS.0.DATEASSESS.COMMONDATE" MONTH="3" DAY="1"
 YEAR="1999" />
 <DATA TAG="VS.0.WEIGHT.PFWT_TC" VALUE="150"
 UNIT="Pound" />
 <DATA TAG="VS.0.TEMPTEXT.TEMPTEXT" VALUE="98.7"
 UNIT="Fahrenheit" />
 <DATA TAG="VS.0.BPREADING.BPREADINGGROUP.SYSTEXT"
 VALUE="130" />
 <DATA TAG="VS.0.BPREADING.BPREADINGGROUP.DIASTEXT"
 VALUE="85" />
</PATIENTDATA>
<!-- DOV form -->
<PATIENTDATA

```

```

 PATIENTINITIALS="ABC" SITEMNEMONIC="PF"
 FORMSETREFNAME="UnschVisit" FORMREFNAME="DOV"
 NEWUNSCHEDVISIT="TRUE">
 <DATA TAG="DOV.0.DOV.DOV" MONTH="2" DAY="2"
 YEAR="1999"/>
</PATIENTDATA>
<!-- VitalSigns form -->
<PATIENTDATA
 PATIENTINITIALS="ABC" SITEMNEMONIC="PF" FORMSETINDEX="2"
 FORMSETREFNAME="UnschVisit" FORMREFNAME="VS">
 <DATA TAG="VS.0.DATEASSESS.COMMONDATE" MONTH="3" DAY="2"
 YEAR="1999"/>
 <DATA TAG="VS.0.WEIGHT.PFWT_TC" VALUE="150"
 UNIT="Pound"/>
 <DATA TAG="VS.0.TEMPTEXT.TEMPTEXT" VALUE="98.7"
 UNIT="Fahrenheit"/>
 <DATA TAG="VS.0.BPREADING.BPREADINGGROUP.SYSTEXT"
 VALUE="130"/>
 <DATA TAG="VS.0.BPREADING.BPREADINGGROUP.DIASTEXT"
 VALUE="85"/>
</PATIENTDATA>

```

### Example 4: Specifying REASONINCOMPLETE

The following XML fragment shows an addition to the Pulse Rhythm item on the Vital Signs (VS) form.

```

<PATIENTDATA
 PATIENTINITIALS="A3" SITEMNEMONIC="PF"
 FORMSETREFNAME="DV1" FORMREFNAME="VS"
 REASONOTHER="test reason">
 <DATA TAG="VS.0.PULSERHYTHM.PULSERHYTHMRADIO"
 COMMENT="irregular REASONINCOMPLETE="NAElement"/>
</PATIENTDATA>

```

### Example 5: Creating a new association instance

The following XML fragment shows the creation of a new association instance.

```

<PATIENTDATA
 PATIENTINITIALS="pjb" SITEMNEMONIC="PF"
 FORMSETREFNAME="Visit6" FORMREFNAME="VS"
 FORMINDEX="2"
 REASONPULLDOWN="3"
 <ASSOCIATION FORMSETREFNAME="Visit1" FORMREFNAME="DEM"
 FORMINDEX="4"/>
</PATIENTDATA>

```

## Updating existing patient clinical data

To modify existing clinical data for a patient at a site, use the EDITPATIENTDATA element with the following attributes:

Attribute	Definition
PATIENTINITIALS	Specifies the initials of the patient that is being enrolled. Either PATIENTINITIALS or PATIENTNUMBER is required.
PATIENTNUMBER	Specifies the patient number of the patient being enrolled. Either PATIENTINITIALS or PATIENTNUMBER is required.
SITEMNEMONIC	Specifies the mnemonic of the site at which the patient is being screened. Either SITEMNEMONIC or SITENAME is required. Either SITEMNEMONIC or SITENAME is required.
SITENAME	Specifies the name of the site at which the patient is enrolled. Either SITEMNEMONIC or SITENAME is required.
FORMSETREFNAME	Specifies the RefName of the visit to which you are importing data.
FORMREFNAME	Specifies the RefName of the CRF to which you are importing data.
REASONPULLDOWN	Specifies the value of a predefined reason for change, as listed in the Reason for Change drop-down list on the Data Value(s) form.
REASONOTHER	Specifies the text of a reason for change, as entered in the Other Reason for Change field on the Data Value(s) form.
REASONINCOMPLETE	This attribute may apply to either the form or item level. The value is one of the values in the radio group control. This attribute will be ignored if the form or item is complete.
CLEARCRF	When true, indicates to ignore all following <Data Tags>.
FORMINDEX	Required for repeating forms. Indicates that a repeating form be updated. Value corresponds to the form instance to update.
ASSOCIATION	Creates or deletes an association instance.
ACTION	ADD or REMOVE; adds or removes an association.

## Editing data in an itemset

To edit data in an itemset on a form, use the following additional attributes:

Attribute	Definition
SECTIONNAME	Specifies the RefName of the section in which the itemset occurs.
ITEMSETNAME	Specifies the RefName of the itemset definition.
ITEMSETINDEX	Number of the itemset line to update. If the ITEMSETINDEX section is blank or has a value of 0, the InForm Data Import utility adds a new itemset line. If ITEMSETINDEX is a number other than 0, the InForm Data Import utility updates the specified line. Note that existing itemset data is not changed, but missing data is filled in. For more information, see <i>Example 2: Creating an itemset row</i> (on page 349).

### Example: Modifying an itemset row

The following XML file fragment shows a modification to the fifth row of itemset data on the Adverse Event (AE) form.

```
<EDITPATIENTDATA
 PATIENTINITIALS="EDT" SITEMNEMONIC="PF"
 FORMSETREFNAME="CommonCRF" FORMREFNAME="AE"
 SECTIONNAME="AE" ITEMSETNAME="AE"
 ITEMSETINDEX="5" REASONOTHER="additional description">
 <DATA TAG="AE.AE.AEDESC.AEDESCTEXT"
 VALUE="Temp spike"/>
</EDITPATIENTDATA>
```



## Updating autocoded data

When you use the InForm Data Import utility to update data that was exported using the autocode feature of the InForm application and coded by an external coding application, the InForm Data Import utility sets the following attribute to TRUE:

Attribute	Definition
AUTOCODEIMPORT	<p>Indicates whether data has been exported using the autocode feature of the InForm application and is being imported after being coded by an external coding application.</p> <p>A value of TRUE indicates that the InForm application must determine whether signatures on the form that is receiving the data must be invalidated. This determination is based on:</p> <ul style="list-style-type: none"> <li>• Visibility of the coded data.</li> <li>• Setting of the INVALIDATIONLEVEL attribute of the SIGNCRF MedML tag that associates a form with a signature group.</li> </ul> <p>For more information, see the <i>Setting Up a Trial with InForm Architect and MedML Guide</i>.</p>

## Editing a comment

You can edit a comment on an item, an item within an itemset, or a form by using the COMMENT attribute:

- To edit a comment on an item, include the COMMENT attribute along with the text of the comment in the DATA element.
- To edit a comment on an itemset, include the COMMENT attribute along with the text of the comment in an EDITPATIENTDATA element that includes the SECTIONNAME and ITEMSETNAME attributes, which signal that the EDITPATIENTDATA element is updating an itemset.
- To edit a comment on a form, include the COMMENT attribute along with the text of the comment in an EDITPATIENTDATA element that does not include SECTIONNAME and ITEMSETNAME attributes.

## Example 1: Changing a data value

The following sample file illustrates a change to the value of the Height item on the DEM form. The Reason for Change is a string other than the predefined Reason for Change strings on the Data Value(s) form. Note that you must enter the UNIT value even if you are not modifying it, or it will be removed.

```
<?xml version="1.0"?>
<CLINICALDATA>
<!-- Demographics form -->
<EDITPATIENTDATA
 PATIENTINITIALS="XYZ"
 SITEMNEMONIC="PF"
 FORMSETREFNAME="Visit1"
 FORMREFNAME="DEM"
 REASONOTHER="received new data">
 <DATA TAG="DEM.0.HEIGHT.PFHT_TC" VALUE="75"
 UNIT="Inches" />
</EDITPATIENTDATA>
</CLINICALDATA>
```

## Example 2: Clearing a data value

The following XML file fragment shows the use of the CLEARVALUE attribute of the DATA element. When you change the selection of a compound radio control, you must clear the value of the original child control.

```
<EDITPATIENTDATA
 PATIENTINITIALS="PE1" SITEMNEMONIC="PF"
 FORMSETREFNAME="Visit1" FORMREFNAME="DEM"
 REASONPULLDOWN="New Information">
 <DATA TAG="RACEGROUP.RACETEXT" CLEARVALUE="TRUE" />
</EDITPATIENTDATA>
```

## Example 3: Specifying REASONINCOMPLETE

The following XML file fragment shows a modification to the Pulse Rhythm item on the Vital Signs (VS) form.

```
<EDITPATIENTDATA
 PATIENTINITIALS="A3" SITEMNEMONIC="PF"
 FORMSETREFNAME="DV1" FORMREFNAME="VS"
 REASONOTHER="test reason">
 <DATA TAG="VS.0.PULSERYTHM.PULSERHYTHMRADIO"
 COMMENT="me too REASONINCOMPLETE="NAElement" />
</EDITPATIENTDATA>
```

## Example 4: Clearing a CRF

The following XML file fragment shows the clearing of a CFR.

```
<EDITPATIENTDATA
 PATIENTINITIALS="DD" SITEMNEMONIC="PF"
 FORMSETREFNAME="VISIT7" FORMREFNAME="PREIM"
 REASONOTHER="clear CFR test">
 CLEARCRF="TRUE"
 <DATA TAG="DEM.0.DTV.DTV_DC" MONTH="4" DAY="22"
 YEAR="2000"
 >
</EDITPATIENTDATA>
```

## Example 5: Deleting an association

The following XML file fragment shows the command to delete an association.

```
<EDITPATIENTDATA
 PATIENTINITIALS="DD" SITEMNEMONIC="PF"
 FORMSETREFNAME="VISIT7" FORMREFNAME="PREIM"
 FORMINDEX="2"
 REASONPULLDOWN="Transcription Error">
 <ASSOCIATION FORMSETREFNAME="Visit6" FORMREFNAME="PE" FORMINDEX="3"
 ACTION="REMOVE"
</EDITPATIENTDATA>
```

## Example 6: Deleting a repeating form

The following XML file fragment shows a command to delete a repeating form.

```
<?xml version="1.0"?>
<CLINICALDATA xmlns="PhaseForward/ImportXML/Inform4">
<EDITPATIENTDATA
 PATIENTINITIALS="XYZ"
 SITEMNEMONIC="PF"
 FORMSETREFNAME="vstAECM"
 FORMREFNAME=" LAE1 "
 FORMINDEX="1"
 <!-- Form Status action:
 DELETE to delete the form
 UNDELETE to restore a deleted form
 -->
 FORMSTATUS="DELETE"
 REASONOTHER="Deleting crf" />
</CLINICALDATA>
```

## Example 7: Restoring a repeating form

The following XML file fragment shows a command to restore (undelete) a repeating form.

```
<?xml version="1.0"?>
<CLINICALDATA xmlns="PhaseForward/ImportXML/Inform4">
<EDITPATIENTDATA
 PATIENTINITIALS="XYZ"
 SITEMNEMONIC="PF"
 FORMSETREFNAME=" vstAECM "
 FORMREFNAME=" LAE1"
 FORMINDEX="1"
 <!-- Form Status action:
 DELETE to delete the form
 UNDELETE to restore a deleted form
 -->
 FORMSTATUS="UNDELETE"
 REASONOTHER="Undeleting CRF" />
</CLINICALDATA>
```

## Transferring a patient record

You can use the InForm user interface to transfer patient information one patient at a time. You can also transfer patients in bulk using the InForm Data Import utility.

You can transfer patients who:

- Change permanent address before completing the trial.
- Have multiple residences throughout the course of the trial.
- Were initially assigned to the wrong sites or to an investigator who is no longer with the trial.

Whether you transfer patients individually, or transfer them in bulk, keep in mind that:

- The InForm application only allows you to transfer patients from one site to another if the trial version at the destination site is the same or greater than the trial version at the current patient site.

To see the trial version for a site, click **Admin > Sites**.

- You can transfer only patients who are fully enrolled; you cannot transfer a patient who is screened but not enrolled or a patient who has failed enrollment.
- You cannot use the patient record transfer feature if you are also using the site filtering feature of the InForm Unplugged application.

To transfer a patient record from one site to another, use the following tags:

- PATIENTSITECHANGE
- NEWSITE
- CURRENTSITE

### PATIENTSITECHANGE

The PATIENTSITECHANGE element identifies each patient to transfer. Use one pair of opening and closing PATIENTSITECHANGE elements for each patient that you want to transfer. The PATIENTSITECHANGE element surrounds all information about a single patient, the current site, and the destination site.

The PATIENTSITECHANGE element has one required attribute:

Attribute	Definition
REASON	A textual description of the reason for the patient transfer.

Within the PATIENTSITECHANGE element, include these elements:

- NEWSITE
- CURRENTSITE

## NEWSITE

The NEWSITE element provides information about the destination site for the patient. Use the NEWSITE element within the opening and closing PATIENTSITECHANGE elements.

The NEWSITE element requires one attribute to identify the destination site. You can use any one of these attributes:

Attribute	Definition
SITEMNEMONIC	Mnemonic for the destination site.
SITENAME	Site name for the destination site.

The NEWSITE tag has one optional attribute:

Attribute	Definition
PATIENTNUMBER	The patient number for the patient at the destination site. Use this attribute only if you need to change the patient number from the one that exists at the current site because a duplicate exists at the destination site.

## CURRENTSITE

The CURRENTSITE element provides information about the current, or originating, site for the patient. Use the CURRENTSITE element within the opening and closing PATIENTSITECHANGE elements.

The CURRENTSITE element requires two attributes: one to identify the current site, and one to identify the patient who is to be transferred.

Use any one of the following attributes to identify the current site:

Attribute	Definition
SITEMNEMONIC	Mnemonic for the destination site.
SITENAME	Site name for the destination site.

Use any one of the following attributes to identify the patient to be transferred:

Attribute	Definition
PATIENTINITIALS	The initials of the patient being transferred. If you use this attribute, and you know that more than one patient at the current site has the specified initials, use the optional DUPLICATEORDER attribute to indicate which patient to transfer.
PATIENTNUMBER	The patient number of the patient being transferred, as it exists on the current site.

Use the following attribute with the CURRENTSITE element if more than one patient at the site has the initials specified in the PATIENTINITIALS element:

Attribute	Definition
DUPLICATEORDER	<p>Specifies which patient should be transferred, if more than one patient at the site has the same initials. When multiple patients have the same initials, the InForm application checks the screening numbers of those patients and transfers the patient whose screening number is in the order specified by the DUPLICATEORDER tag.</p> <p>Use this attribute only if you have used PATIENTINITIALS to identify the patient.</p>

## Example

This example shows the elements that are used in a MedML file to transfer several patients. Note that each of the pairs of PATIENTSITECHANGE elements defines current and destination site information for one patient.

```
<?xml version="1.0"?>
<CLINICALDATA>
 <PATIENTSITECHANGE REASON="new patient address">
 <NEWSITE SITEMNEMONIC="MCLEAN"/>
 <CURRENTSITE SITEMNEMONIC="PF" PATIENTNUMBER="1003"/>
 </PATIENTSITECHANGE>
<!--For patient 1001 the file specifies a new patient number because a patient already exists at the
McLean Hospital site with the same patient number.-->
 <PATIENTSITECHANGE REASON="new patient address">
 <NEWSITE SITENAME="McLean Hospital" PATIENTNUMBER="1002"/>
 <CURRENTSITE SITEMNEMONIC="PF" PATIENTNUMBER="1001"/>
 </PATIENTSITECHANGE>
<!--The DUPLICATEORDER tag indicates that patient DDD is the second patient with those
initials to be screened at the Oracle site.-->
 <PATIENTSITECHANGE REASON="new patient address">
 <NEWSITE SITEMNEMONIC="MCLEAN"/>
 <CURRENTSITE SITENAME="Oracle" PATIENTINITIALS="DDD"
 DUPLICATEORDER="2"/>
 </PATIENTSITECHANGE>
</CLINICALDATA>
```

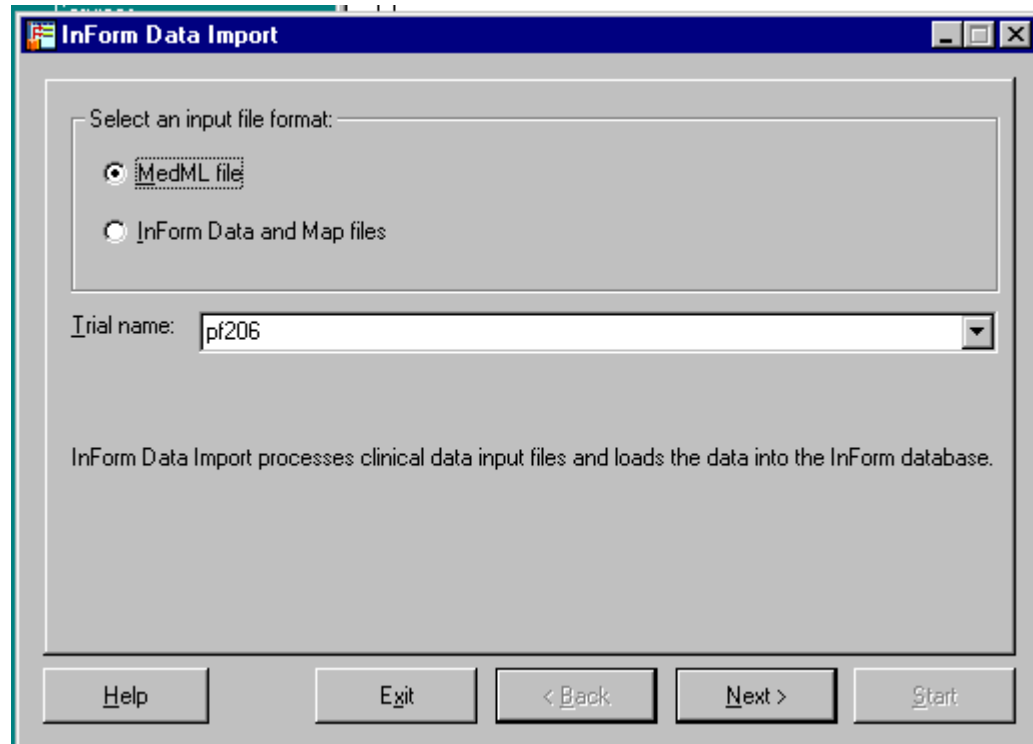
## Running the import using the XML import file

To import an XML file into the database and submit it through the InForm application server:

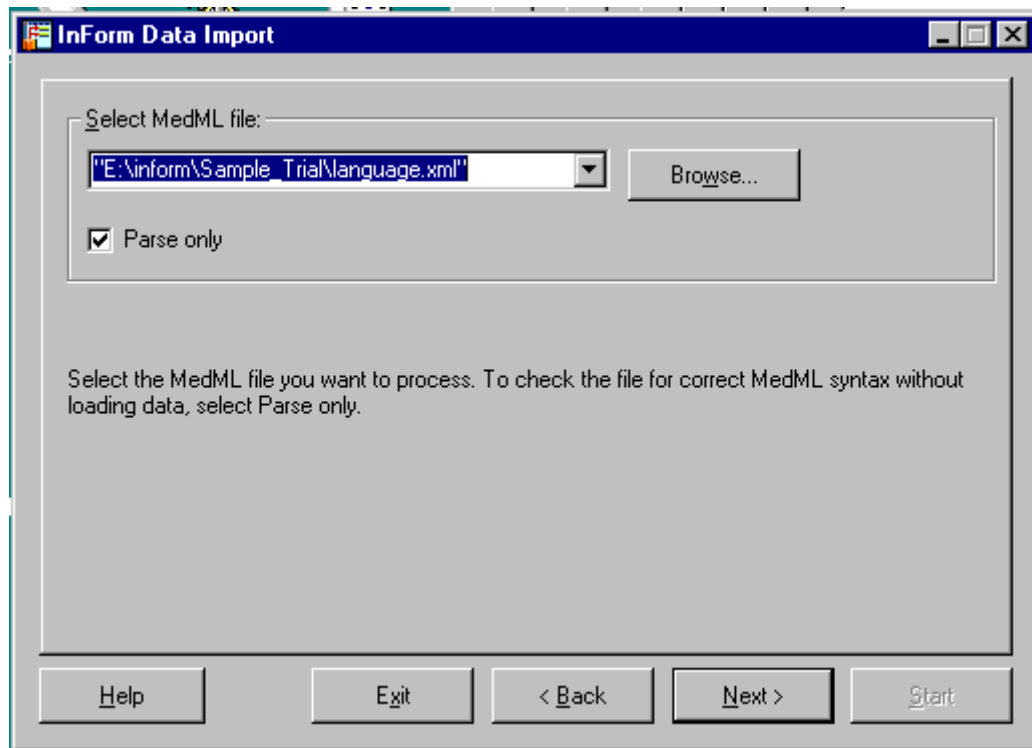
**Note:** To import data into the database, the server must be running before you start the InForm Data Import utility.

- 1 Click **Start > Programs > Oracle Health Science > InForm 4.6.5 > InForm Data Import utility**.

The InForm Data Import utility main window appears.



- 2 Select the **MedML file** radio button.
- 3 In the **Trial Name** field, type or select the name of the trial into which to import the MedML file. The last 10 trials you accessed are stored in the drop-down list.
- 4 Click **Next**.



- 5 In the **Select MedML file** field, type the full path name of the XML file to import, or click **Browse** and locate the file. Any of the following options are valid:
  - The name of one XML file.
  - The name of multiple XML files, each separated by a space.
  - The name of a response file that contains multiple XML files, one per row, in the following format:  
@filename
- 6 Select **Parse Only** to run the import without actually importing data into the database. Use this function to test the syntax of your MedML file before you import it into the database.
- 7 Click **Next**.



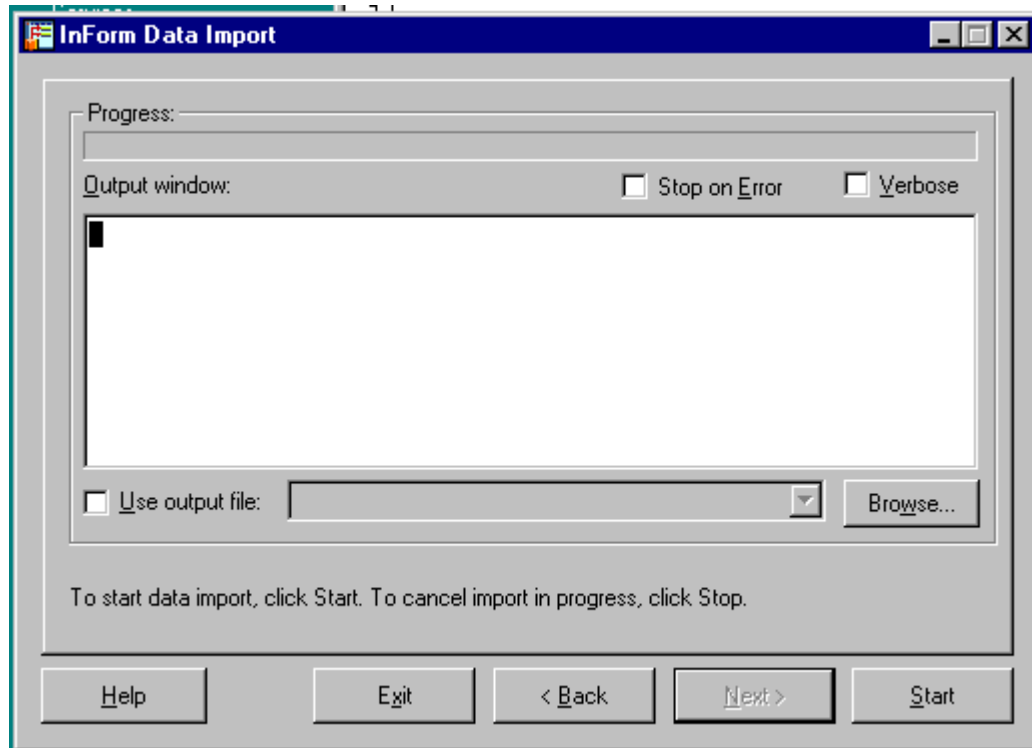
A dialog appears and requests your InForm name and password.

- 8 In the **Name** field, type the name of an InForm user who has the appropriate rights for the data you are importing:

To import data that matches this InForm system activity	User needs these rights
Add patient clinical data	Enter Data into a CRF
Update patient clinical data	Edit Data on a CRF

- 9 In the **Password** field, type the user password, then click **Next**.

The Summary window appears.



**Note:** If you selected **Parse Only**, you do not need to specify a name and password.

- 10 Optionally, select any of the following:
  - **Stop on Error**—To instruct the InForm Data Import utility to stop if it encounters an error.
  - **Verbose**—To instruct the InForm Data Import utility to generate detailed messages as it processes the file.
  - **Use output file**—Specify the filename to save the output file as a text file.
- 11 Click **Start**.  
The InForm Data Import utility processes the import file, writes messages to the message area and the output file, if specified, and adds or updates data in the database.
- 12 Close the InForm Data Import utility. To view the data that you imported, you must stop and restart the trial.

# Importing coded items

## Creating an autocode import file

The import file that is used in the autocoding process must originate from the InForm Data Export utility, and must have been run previously on the same server. The source and destination must be the same machine.

This example is a sample autocode file format that was generated by the InForm Data Export utility:

```
<AUTOCODEDATA>

<AUTOCODESET CODESETTYPE="AE">

<AUTOCODE VERBATIM="Headache" CODEVALUE="HED">
<CODEMAP
CODEMAPKEY="840C35C7-BD2D-4C3A-AA4E-EF396CDCD44B"
SOURCEID="FDBFA029-B175-47D4-B16D-D6618E90D8A0"/>
</AUTOCODE>

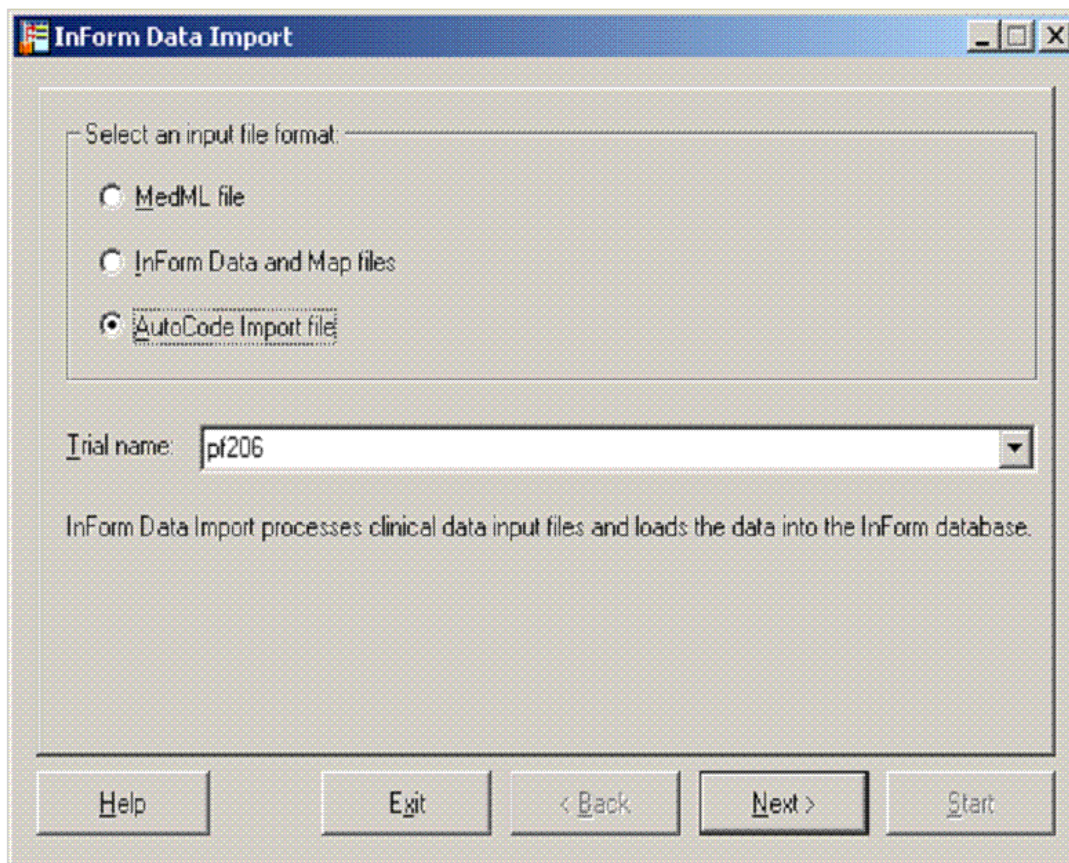
<AUTOCODE VERBATIM="Vomiting" CODEVALUE="VMT">
<CODEMAP
CODEMAPKEY="5C05A861-91F5-4DDE-87C3-CB04634A7520"
SOURCEID="B18EF874-8503-4058-B364-49BE41744003"/>
</AUTOCODE>

</AUTOCODESET>
</AUTOCODEDATA>
```

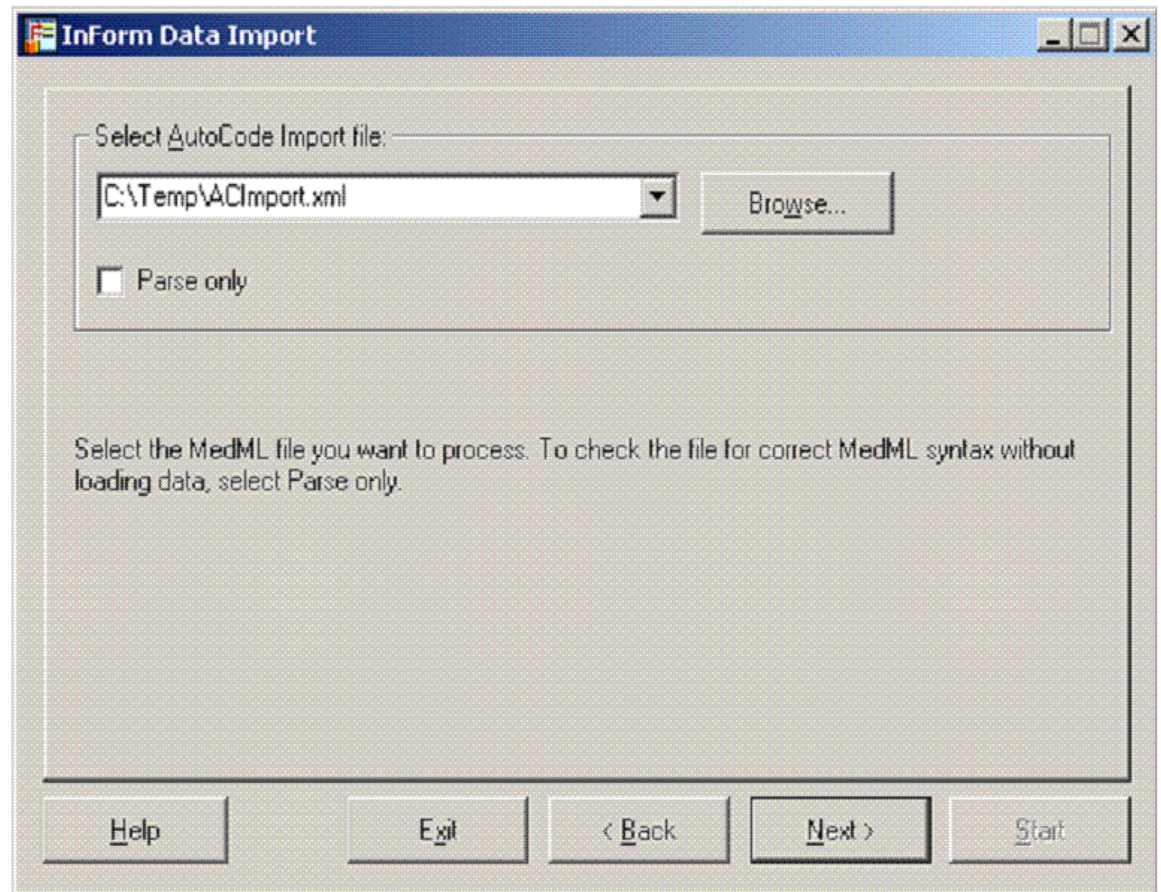
## Running the import using an autocode import file

To run the import:

- 1 Select **Start > Programs > Oracle Health Science > InForm 4.6.5 > InForm Data Import utility**.
- 2 When the InForm Data Import utility window appears, select the **AutoCode Import file** radio button.



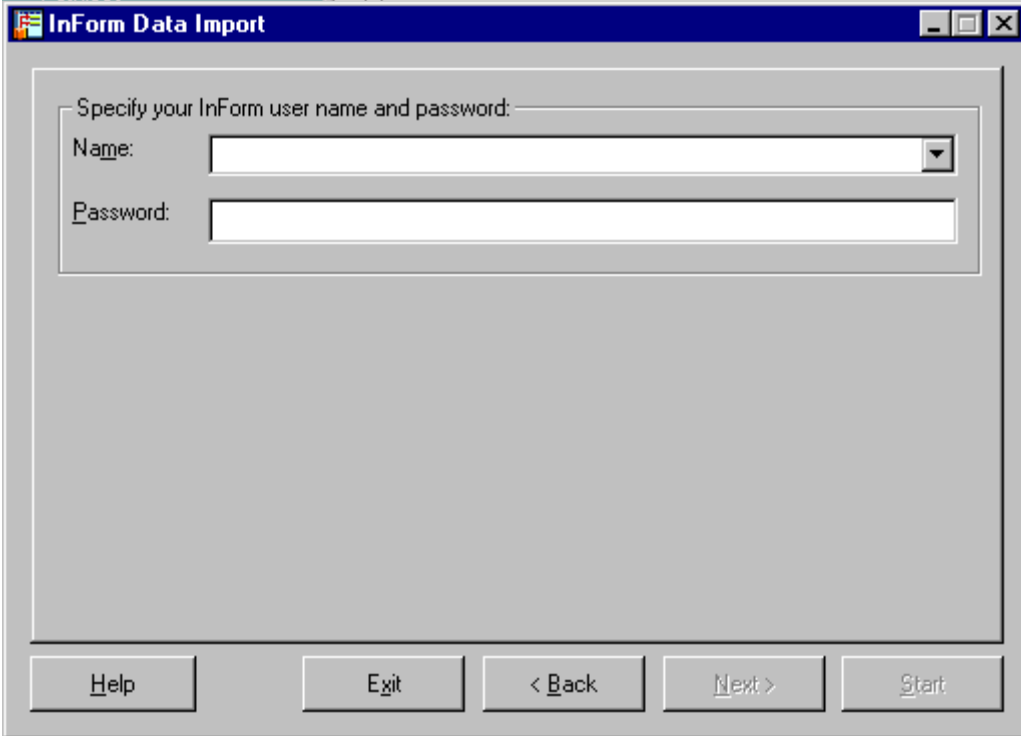
- 3 Select a trial.
- 4 Click **Next**.



- 5 In the **Select AutoCode Import file** field, type the full path name of the autocode XML file to import, or click **Browse** and locate an existing autocode XML file.

**Note:** This input file must originate from the InForm Data Export utility from the same source and destination machine.

A dialog appears and requests your InForm name and password.



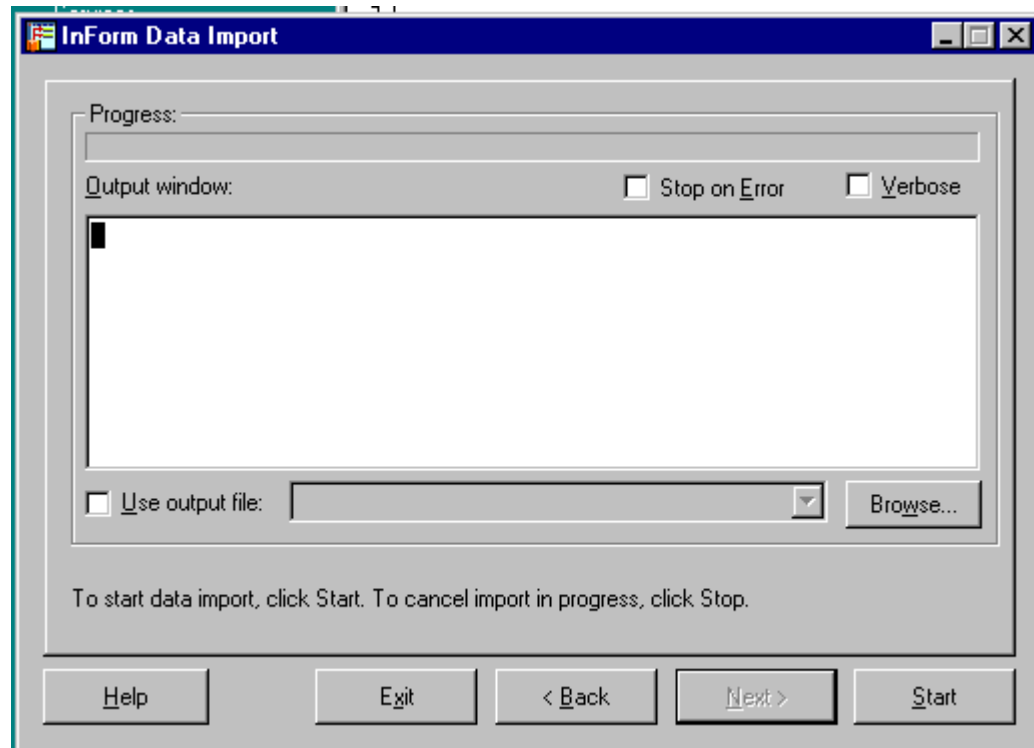
The screenshot shows a standard Windows-style dialog box titled "InForm Data Import". The main area contains the instruction "Specify your InForm user name and password:". Below this, there are two input fields: "Name:" followed by a text box with a dropdown arrow, and "Password:" followed by a text box. At the bottom of the dialog, there are five buttons: "Help", "Exit", "< Back", "Next >", and "Start".

- 6 In the **Name** field, type the name of an InForm user who has the appropriate rights for the data you are importing:

To import data that matches this InForm system activity	User needs these rights
Add patient clinical data	Enter Data into a CRF
Update patient clinical data	Edit Data on a CRF

- 7 In the **Password** field, type the user password, then click **Next**.

The Summary window appears.



- 8 Optionally, select any of the following:
  - **Stop on Error**—To instruct the InForm Data Import utility to stop if it encounters an error.
  - **Verbose**—To instruct the InForm Data Import utility to generate detailed messages as it processes the file.
  - **Use output file**—Specify the filename to save the output file as a text file.
- 9 Click **Start**.

The InForm Data Import utility processes the import file, writes messages to the message area and the output file, if specified, and adds or updates data in the database.
- 10 Close the InForm Data Import utility. To view the data that you imported, you must stop and restart the trial.

## Date and time validation

The InForm Data Import utility performs the following validation checks for datetime data:

Validation check	Description
DISPLAY attribute of CRF datetime control.	The MedML DISPLAY attribute must be set to True for the components of the CRF datetime control into which a date, time, or datetime item is being imported. The InForm Data Import utility ignores date and time components for which the DISPLAY attribute is False, and it does not import them.
Year value of import field.	If the imported year is outside the range allowed in the CRF datetime control, the InForm Data Import utility issues an error.
Day value of import field.	If the Year and Month values are valid, the InForm Data Import utility checks for a valid day value.
REQUIRED attribute of CRF datetime control.	If any datetime component is coded as REQUIRED in the CRF datetime control and is missing in the import file, the InForm Data Import utility issues an error.
UNKNOWN attribute of CRF datetime control.	If an imported datetime component is coded UNK, but the UNKNOWN attribute of the CRF datetime control is not enabled, the InForm Data Import utility issues an error.
Consistency checking.	If consistency checking is enabled in the CRF datetime control, the InForm Data Import utility issues an error if a datetime component is present in the import file without a higher component, or an UNK value, in the following sequence: <code>ss:mm:hh dd/mm/yy</code> For example, if the imported datetime includes a Day field without a Month, the InForm Data Import utility issues an error.
Hour and Minute values of import field.	The values for both Hour and Minute must be within valid ranges or coded as UNK.



## Additional InForm Data Import utility attributes

Attribute	Description	Available values
CLEARCRF	Indicates whether to clear the specified form.	TRUE, FALSE
COMMONFORM	Indicates whether a form is a common CRF.	TRUE, FALSE
DELETEITEMSET	Indicates whether to delete an itemset.	TRUE, FALSE
FORMSTATUS	Modifies the status of a form.	<ul style="list-style-type: none"> <li>• FREEZE</li> <li>• UNFREEZE</li> <li>• LOCK</li> <li>• UNLOCK</li> <li>• DELETE</li> <li>• UNDELETE</li> </ul>
OVERRIDE	Specifies whether a patient screening or enrollment has been overridden.	TRUE, FALSE
UNDELETEITEMSET	Indicates whether to undelete an itemset.	TRUE, FALSE

## Running the InForm Data Import utility from the command line

While you are creating or editing your map file, use the InForm Data Import utility in interactive mode. After your map file is stable, it may be more convenient to run the utility from the command line, or to insert the import command in a batch file that is scheduled to run periodically.

**Note:** Oracle strongly recommends that you run the InForm Data Import utility through the PFConsole utility.

A space is required between the parameter name and its value.

```
PFImport [-?] [-autorun] [-verbose] [-parse] [-trial <trialname>] [-errstop] -
norules [[-user <username>] [-password <password>] [@<rsp_file>] [-xml
<xml_file>...<xml_file>]] [-AUTOCODE] [-template <map_file>] [-import
<data_file>]
```

Use the following command line parameters. You must include a space between the parameter name and its value.

Parameter	Variable	Description
PFImport		Starts the InForm Data Import utility.
-?		Displays command help.
-autorun		Runs the InForm Data Import utility in a command window.
-verbose		Indicates that the InForm Data Import utility will write detailed messages as it processes the input file.
-parse		Indicates that the import will run without actually importing data into the database. Use this function to test the syntax of your MedML file before it is imported into the database.
-trial	<i>trialname</i>	Specifies the trial into which you are importing data. Use the full pathname of the trial.
-errstop		When the errstop parameter is specified, the InForm Data Import utility will stop processing when it encounters an error. When errstop is not specified, the tag containing the error is skipped and the import continues with the next data tag in the file.
-norules		Indicates that rules will not run during the import. This parameter is required.
-user	<i>username</i>	The name of an InForm user who has the appropriate rights for the data you are importing.
-password	<i>password</i>	The password associated with the user specified by the user parameter.

Parameter	Variable	Description
-validate		Checks to make sure that all required XML tags exist and the specified control paths can be found, without loading data. Optionally, use this parameter to validate an XML file before importing it.
-unit		Converts imported units to base units and stores both imported and base values. To use this parameter, the Rule Manager (InFormRule MTS package) must be running.
@	<i>RSP_file</i>	Indicates that you are submitting a response file (RSP extension) that contains the names of the XML files to process. Specify the full pathname of the response file. Use this option or the -xml option or the -template and -import options.
-xml	<i>xml_file</i>	Indicates that you are submitting one or more XML files to process. If you are submitting multiple XML files, separate them with a space. Use this option, the @ option, or the -template and -import options.
-AUTOCODE		Indicates that the specified XML file or files contain autocode data.
-template	<i>map_file</i>	If you are using the data and map file option, indicates that the next parameter is the map file name. Specify the full pathname of the map file. Use this option or the @ option or the -xml option. If you specify a map file, you must also specify an import file with the -import option.
-import	<i>data_file</i>	Indicates that the next parameter is the import file name. Specify the full pathname of the data file. Use this option or the @ option or the -xml option. If you specify an import file, you must also specify a map file with the -template option.
-output	<i>output_file</i>	Indicates that you want to save the output in a text file. Specify the full path and file name you want to give to the file. When you use this parameter, output messages are not displayed in the Output window.

## Enhancing your data import

### Log off of the InForm application server

Log off of the InForm application server before running the import. You can keep the server running.

### Stop the WWW Publishing Service

Stop the WWW Publishing Service to prevent others from entering data as you perform an import. Make sure you turn the WWW Publishing Service back on when your import is complete.

### Run the Oracle update statistics script

Run the Oracle update statistics script immediately after you import files. A single import can enter in as much data as you might manually enter in 6 weeks.

### Change the home page

Before you start the import, change the home page of the trial and warn users that they may experience slowness in the system while you are running the import.

### Organize by patient

If possible, sort the data you are importing by patient ID.

## CHAPTER 5

# InForm Data Export utility

### In this chapter

Overview of the InForm Data Export utility.....	374
Running the InForm Data Export utility.....	375
Exporting coded controls.....	377
Exporting data into a CDD.....	381
Exporting Name Value Pairs .....	384
Exporting data in Oracle Clinical format.....	387
Running the InForm Data Export utility from the command line.....	392

# Overview of the InForm Data Export utility

## InForm Data Export utility output options

The InForm Data Export utility exports data from the InForm database in several formats. The following output options are available in the InForm Data Export utility:

- **AutoCode**—Facilitates the use of external coding tools to code items.
- **CDD**—Exports data into a Customer-Defined Database.
- **Name Value Pairs**—Creates a pipe-delimited file that consists of data path names and data values.
- **Oracle® Clinical**—Creates a text file that can be used to import information into an Oracle Clinical database.

**Note:** Use of the InForm Data Export utility is limited to trials with fewer than 100,000 patients.

## Running the InForm Data Export utility

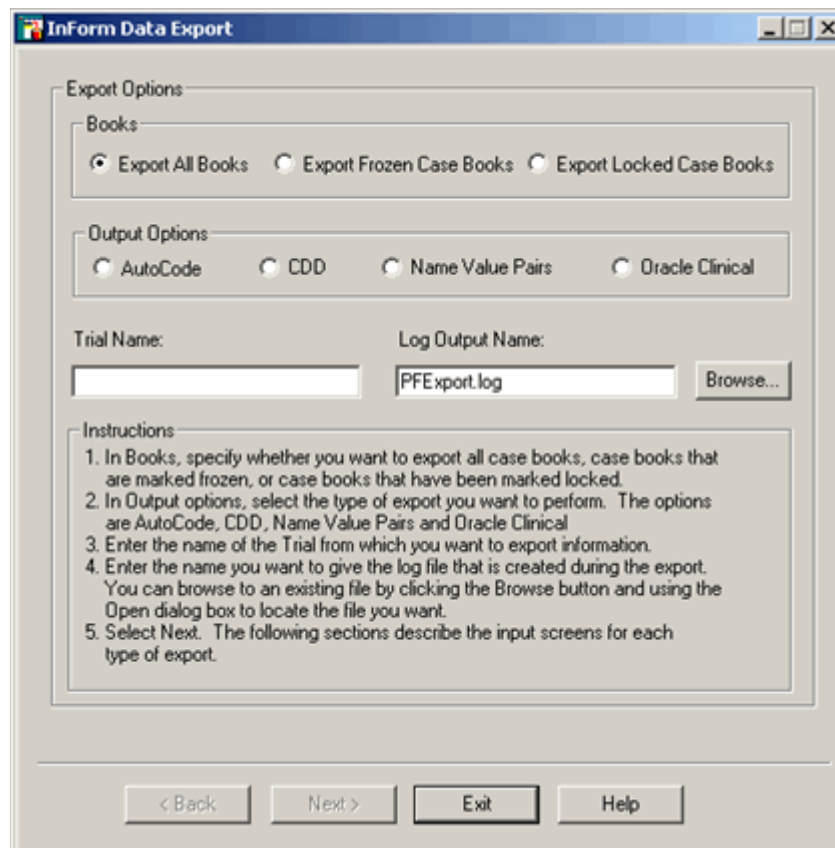
All of the InForm Data Export utility options run under the same executable.

**Note:** To run the InForm Data Export utility, the InForm application server does not need to be running.

To run the InForm Data Export utility:

- 1 Click **Start > Programs > Oracle Health Science > InForm 4.6.5 > InForm Data Export utility**.

The InForm Data Export utility main window appears.



- 2 In the **Books** section, select which cases to export.
- 3 In the **Output Options** section, select the format to use for the export. The options are:
  - AutoCode
  - CDD
  - Name Value Pairs
  - Oracle Clinical

For more information, see *InForm Data Export utility output options* (on page 374).

- 4 In the **Trial Name** field, type the name of the trial from which you want to export information.

- 5 In the **Log Output Name** field, type the path and file name to give the log file that is created during the export, or click **Browse** and locate the file to use as the log file.
- 6 Click **Next**.

The following sections describe the input screens for each type of export:

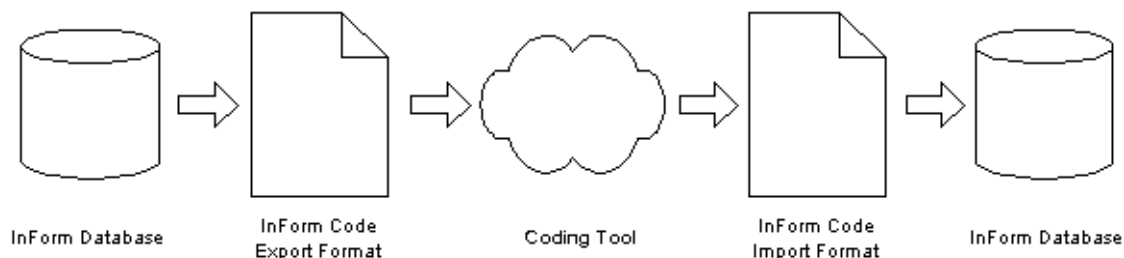
- *Exporting coded controls* (on page 377).
- Exporting data into a CDD.
- *Exporting Name Value Pairs* (on page 384).
- *Exporting data in Oracle Clinical format* (on page 387).



## Exporting coded controls

The InForm Data Export utility can export source item values (Verbatim values) and their corresponding destination targets (Code Value values) with information that allows the values to be mapped back during import.

The coding process uses several steps, as shown in the flowchart:



For more information, see *Importing coded items* (on page 363).

For more information on coding the controls through the InForm Architect application, see the *Setting Up a Trial with InForm Architect and MedML Guide*.

## Required elements

The following data elements are required during export:

- SOURCE (VERBATIM) and TARGET (CODE VALUE) control mapping information.
- The full path of the SOURCE controls that are carried by the CODEMAPKEY and SOURCEID attributes, and are exported as GUIDs.
- The original value of the source control. This is also required during import in case the value of the source control is changed between export and import.
- The value of the code (destination) control.

## Example

Example of the XML code:

```

<AUTOCODEDATA>
<AUTOCODESET CODESETTYPE="AE">
<AUTOCODE VERBATIM="Headache" CODEVALUE="">
 <CODEMAP
 CODEMAPKEY="840C35C7-BD2D-4C3A-AA4E-EF396CDCD44B"
 SOURCEID="FDBFA029-B175-47D4-B16D-D6618E90D8A0"/>
</AUTOCODE>
<AUTOCODE VERBATIM="Vomiting" CODEVALUE="">
 <CODEMAP
 CODEMAPKEY="5C05A861-91F5-4DDE-87C3-CB04634A7520"
 SOURCEID="B18EF874-8503-4058-B364-49BE41744003"/>
</AUTOCODE>
</AUTOCODESET>
</AUTOCODEDATA>

```

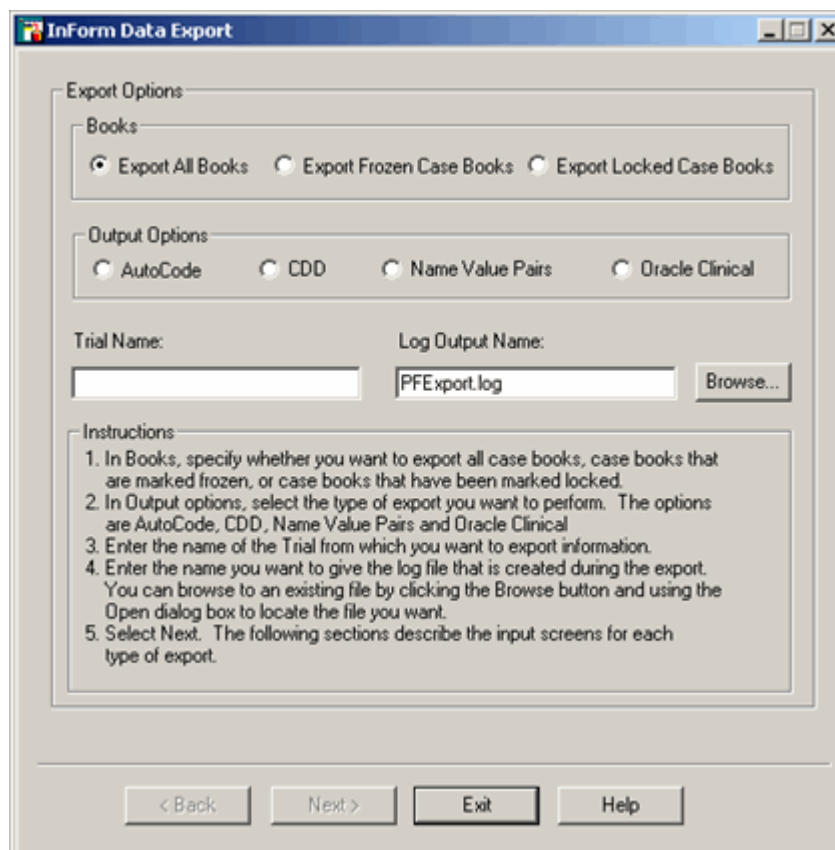
This XML is used for both export and import. In this sample, the CODEVALUE tag is empty, which indicates that the verbatim is not yet coded. During export, the tag could have a value if the data that is exported has coded values that need to be recoded. After import, the CODEVALUE tag will have values.

The complete set of items to be coded is wrapped in an AUTOCODESET object with the qualifier of *type*. The type is specified at design time, in order to allow autocode fields to be grouped more easily.

## Running the export for AutoCode items

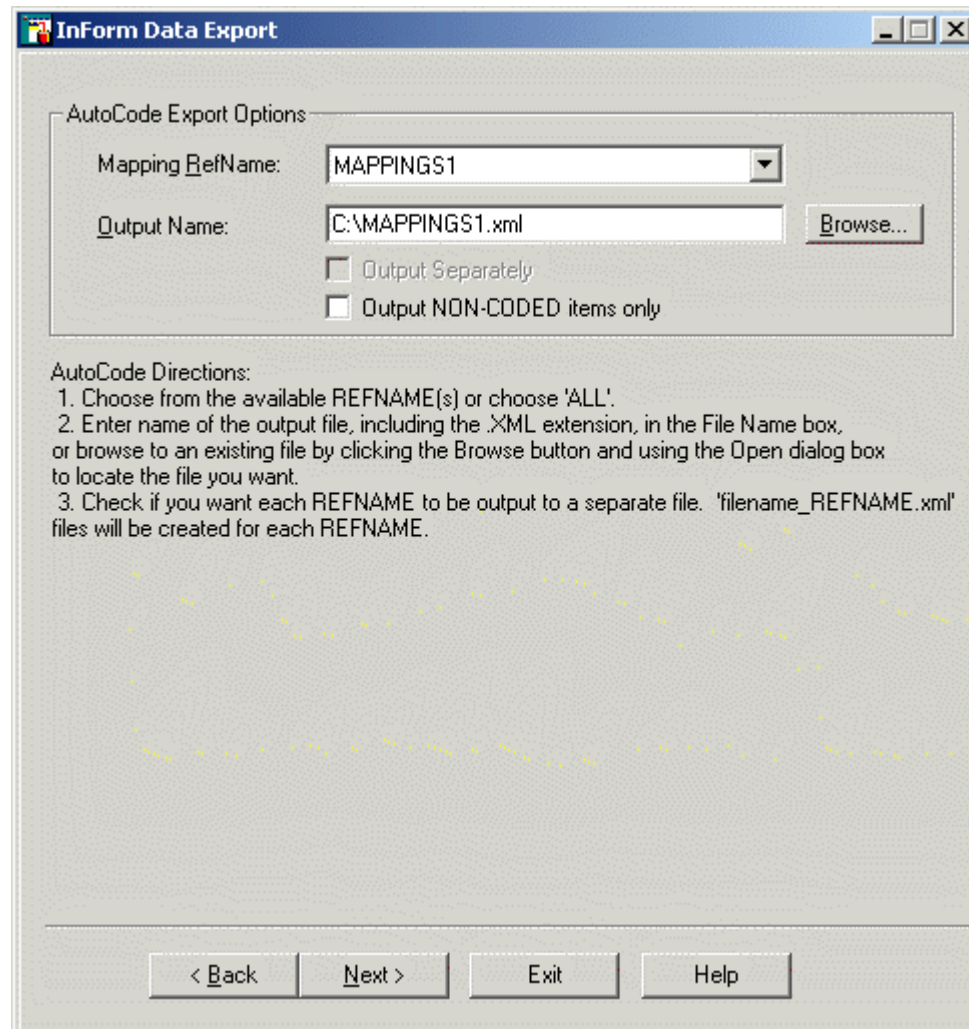
To run the export:

- 1 Click **Start > Programs > Oracle Health Science > InForm 4.6.5 > InForm Data Export utility**.



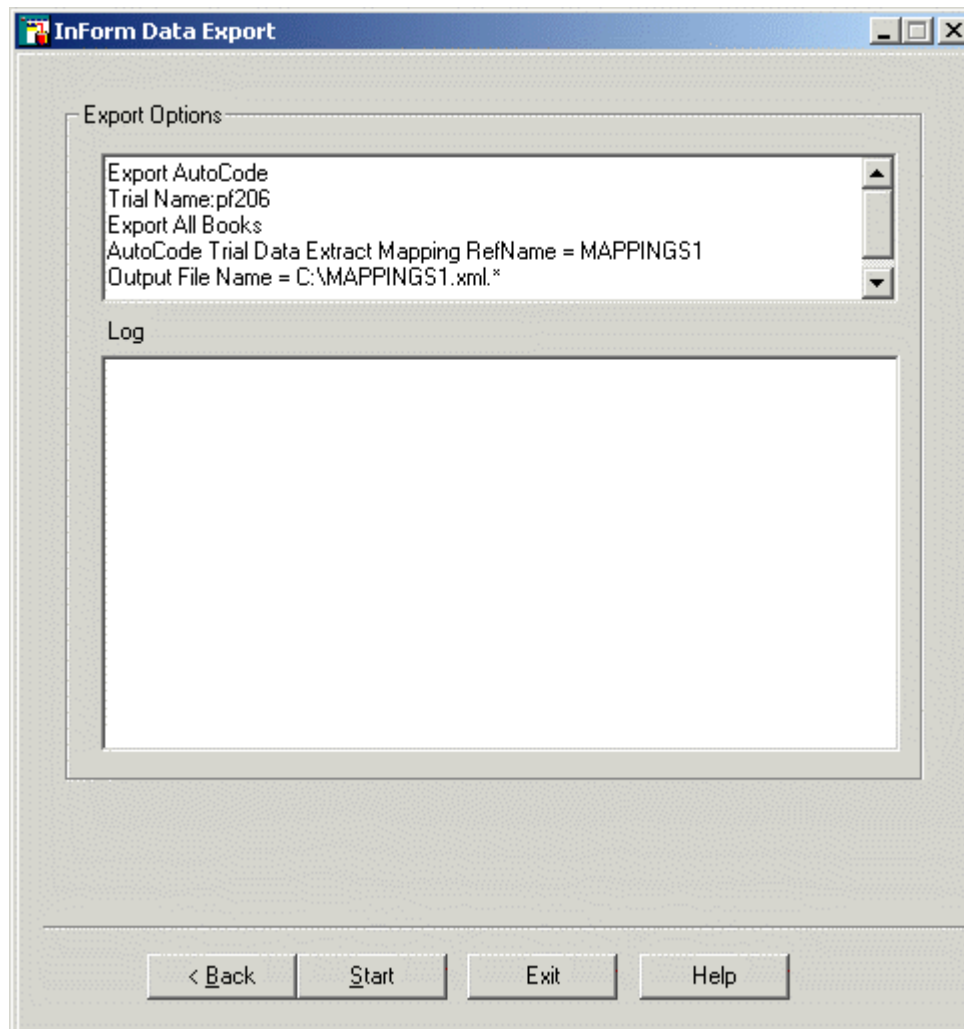
- 2 In the Export Options window:
  - In the **Books** section, select which case books to export.
  - In the **Output Options** section, select **Autocode**.
  - Specify the **Trial Name** and the **Log Output Name**, or click **Browse** and locate the log output name.
- 3 Click **Next**.

The AutoCode Export Options window appears.



- 4 In the **Mapping RefName** drop-down list, select an autocode target set(s). To export all target sets, select **All**.
- 5 In the **Output Name** field, type the name of the output file, including the XML extension, or click **Browse** and locate the file.
- 6 Select **Output Separately** to export each mapping RefName to a separate file. A file with the name OutputName\_REFNAME.xml will be created for each mapping RefName that is defined in the system.
- 7 Select **Output NON-CODED items only** to output only non-coded items.
- 8 Click **Next**.

The **Log** section appears below the **Export Options** section.



- 9 Click **Start**.

# Exporting data into a CDD

## Overview

The **CDD** output option reads the CDD mapping parameters and all of the data in the InForm database, then populates the CDD in the following order:

- Site data
- Comments
- Patient information
- CRF data

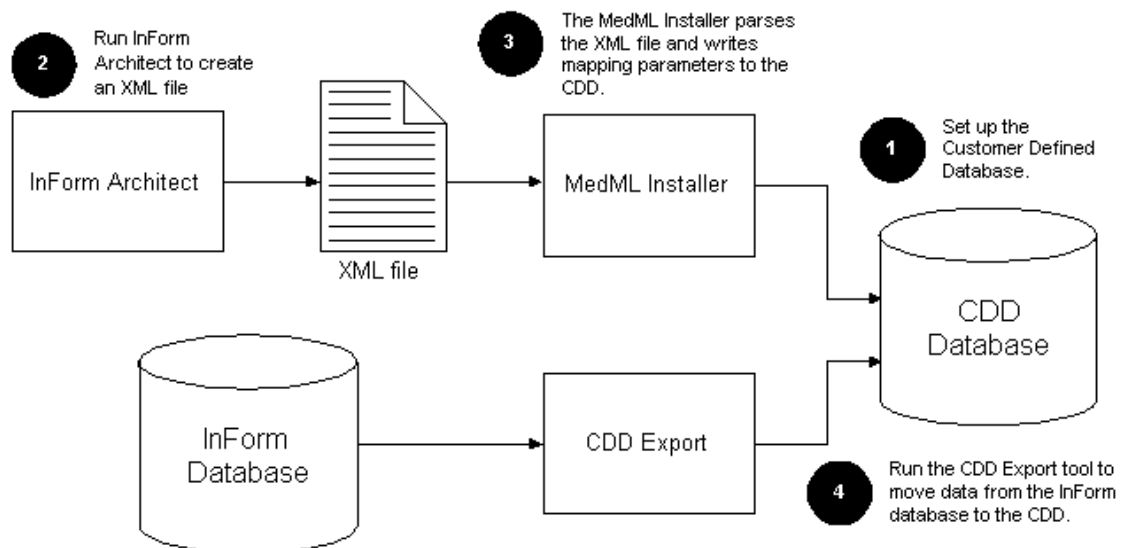
This is useful for repopulating the database when the CDD definition changes after a trial starts.

## Moving data to a CDD

To move data from the InForm trial database to a CDD:

- 1 Use the InForm Architect application to generate a CDD definition that includes parameters that specify how to map data from the InForm database to the CDD. Save the definition to an XML file. For more information, see the *Setting Up a Trial with InForm Architect and MedML Guide*.
- 2 Run the MedML Installer utility to parse the XML file and write the mapping parameters to the InForm database.
- 3 Run the InForm Data Export utility using the **CDD** output option to read the data in the InForm database and populate the CDD database. For more information, see *Running the export for CDD data* (on page 382).

The flowchart shows the export process:



## Running the export for CDD data

**Note:** Before you run the InForm Data Export utility for CDD data, you should first stop the InForm application server.

To export data to a CDD:

- 1 Back up the existing CDD.
- 2 In the **Books** section, select one of the following:
  - Export All Books
  - Export Frozen Case Books
  - Export Locked Case Books
- 3 In the **Output Options** section, click **CDD**.
- 4 In the **Trial Name** field, type the trial name.
- 5 In the **Log Output Name** field, type the name of the log file to which to save the export, or click **Browse** to locate the file.

The CDD Export Options dialog box appears.

- 6 On the CDD Export Options window, type the name of the ODBC DSN that is defined for the CDD to which to output data. For more information about creating a new DSN, see *Creating a new DSN for the export* (on page 383).
- 7 In the **User Name** and **Password** fields, type the user name and password that is used to access the CDD.
- 8 Select the **Create New Schema** checkbox to drop the current database and create a new schema based on the RefName associated with the DSN, and type the name of the tablespace in which to put the new schema in the **Table Space** field.
- 9 Click **Next**.

The Export Summary screen appears.

**WARNING:** Make sure that all the information is correct before you proceed to the next step. When you run the InForm Data Export utility, all data in the current database will be lost.

- 10 Click **Finish**.

The InForm Data Export utility begins to populate the CDD and displays messages to indicate its progress.

## Creating a new DSN for the export

To create a new DSN:

- 1 On the CDD Export Options window, in the **DSN List** field, type the new name of the ODBC DSN for the CDD to which to output data.
- 2 Type the user name and password for accessing the CDD.
- 3 Click **Create DSN**.

The CDD Data Source window appears. Your new DSN appears in the first field.

The screenshot shows a dialog box titled "CDD Data Source". It has three input fields: "CDD DSN:" containing "NEW CDD", "CDD Refname:" with a dropdown menu showing "DemoGraphics", and "Database Server:" containing "OraDB". At the bottom are "Cancel" and "OK" buttons.

- 4 From the **CDD RefName** drop-down list, select the RefName to associate with the new DSN.
- 5 In the **Database Server** field, type the name of the server to use.
- 6 Click **OK**.

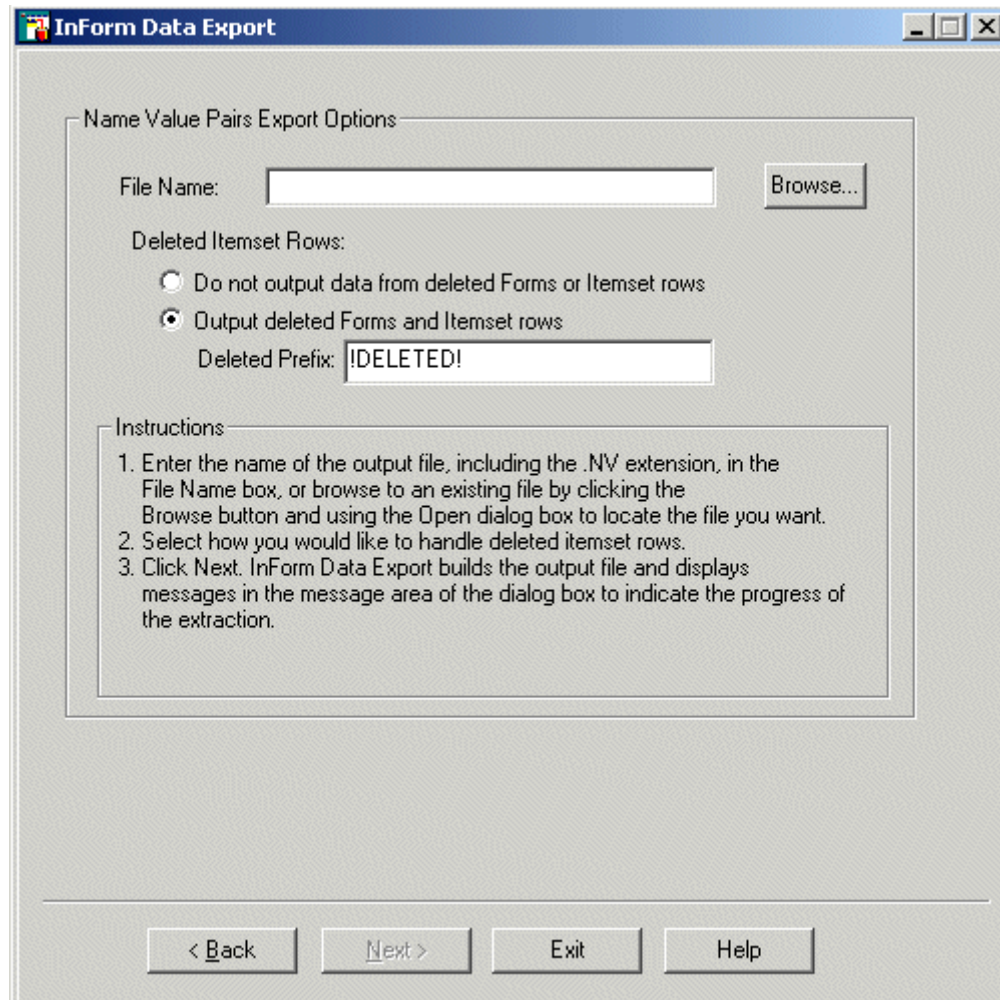
The CDD Export Options window reappears, and the new DSN appears in the drop-down list.

**Note:** If you create a new DSN for the export, the new DSN is valid only in the InForm Data Export utility session you are in when you create it. It will be removed when you close the InForm Data Export utility. To create a DSN to use outside of the InForm Data Export utility use the ODBC Manager or PFAAdmin CONFIG CDD Setup commands. For more information on PFAAdmin commands, see the *Installation and Configuration Guide*.

## Exporting Name Value Pairs

The **Name Value** output option exports data from the InForm database into a plain text file. It is useful for extracting data for conversion or analysis.

**Note:** The Name Value output option exports only named values that occur in patient data; it does not export values from the Reg Docs or Visit Reports forms.



To export a Name Value data file:

- 1 In the **Books** section, select one of the following:
  - Export All Books
  - Export Frozen Case Books
  - Export Locked Case Books
- 2 In the **Output Options** section, click **Name Value Pairs**.
- 3 In the **Trial Name** field, type the trial name.



- 4 In the **Log Output Name** field, type the name of the log file to which to save the export, or click **Browse** to locate the file.

The Name Value Pairs Export Options dialog box appears.

- 5 In the **File Name** field, type the name of the output file, including the NV extension, or click **Browse** to locate the file.
- 6 In the **Deleted Itemset Rows** section, select one of the following:
  - Select **Do not output data from deleted Forms or Itemset rows** to instruct the InForm Data Export utility not to output data from deleted forms or itemset rows.
  - Select **Output Deleted Forms and Itemset Rows** to export data from deleted forms or itemset rows.
  - Then, in the **Deleted Prefix** box, specify the prefix to add to each deleted itemset row that the InForm Data Export utility exports.
- 7 Click **Next**.

The InForm Data Export utility builds the output file and displays messages to indicate the progress of the extraction.

## Output file format

In the output file that is created by running the export for name value pairs, fields are separated by pipes, and inapplicable fields are left blank, using the following data:

Name Value field	Description
Patient identifier	Patient initials followed by patient number in parentheses.
RefName path	Path of RefNames in the following order: <ul style="list-style-type: none"> <li>• Visit.</li> <li>• Repeating formset index, which is used to indicate which instance of a recurring unscheduled visit is referenced. If the visit is not an unscheduled visit, the value is 1.000.</li> <li>• Form.</li> <li>• Repeating form index.</li> <li>• Section.</li> <li>• Itemset.</li> <li>• Itemset index, which is used to indicate which row of an itemset is referenced.</li> <li>• Item.</li> <li>• Control.</li> </ul>
Normalized value	Value after conversion into the base units that are specified in the database.
Entered value	Value as entered for the control.

## Output file format for associated forms

In the output file that is created by running the export for associated forms, fields are separated by pipes, and inapplicable fields are left blank, using the following data:

Associated form field	Description
Patient identifier	Patient initials followed by patient number in parentheses.
RefName path	Path of RefNames in the following order. <ul style="list-style-type: none"> <li>• Visit.</li> <li>• Repeating formset index, which is used to indicate which instance of a recurring unscheduled visit is referenced. If the visit is not an unscheduled visit, the value is 1.000.</li> <li>• Form.</li> <li>• Repeating form index.</li> </ul> <p>The path for the first form appears in this order, and the path for the associated form appears after the first form, surrounded by double quotation marks.</p>
Normalized value	Value after conversion into the base units that are specified in the database.
Entered value	Value as entered for the control.

## Example

The following export file fragment illustrates some of the data exported for the patient PA1(1).

```
PA1(1) | Visit1 | 1 | EYE | VISION | 0 | LNEAR.EYEGROUP | |
PA1(1) | Visit1 | 1 | EYE | VISION | 0 | LNEAR.EYEGROUP.EYE_TC | 20.000000 | 20
PA1(1) | Visit1 | 1 | EYE | VISION | 0 | LNEAR.EYEGROUP.UNEYE_TC | 20.000000 | 20
PA1(1) | Visit1 | 1 | EYE | VISION | 0 | RNEAR.EYEGROUP | |
PA1(1) | Visit1 | 1 | EYE | VISION | 0 | RNEAR.EYEGROUP.EYE_TC | 20.000000 | 20
PA1(1) | Visit1 | 1 | EYE | VISION | 0 | RNEAR.EYEGROUP.UNEYE_TC | 20.000000 | 20
PA1(1) | Visit1 | 1 | EYE | VISION | 0 | LDIST.EYEGROUP | |
PA1(1) | Visit1 | 1 | EYE | VISION | 0 | LDIST.EYEGROUP.EYE_TC | 20.000000 | 20
PA1(1) | Visit1 | 1 | EYE | VISION | 0 | LDIST.EYEGROUP.UNEYE_TC | 20.000000 | 20
PA1(1) | Visit1 | 1 | EYE | VISION | 0 | RDIST.EYEGROUP | |
PA1(1) | Visit1 | 1 | EYE | VISION | 0 | RDIST.EYEGROUP.EYE_TC | 20.000000 | 20
PA1(1) | Visit1 | 1 | EYE | VISION | 0 | RDIST.EYEGROUP.UNEYE_TC | 20.000000 | 20
PA1(1) | Visit1 | 1 | EYE | RETINAL | 0 | RWOOL.WOOLGROUP | | N
PA1(1) | Visit1 | 1 | EYE | RETINAL | 0 | LWOOL.WOOLGROUP | | N
PA1(1) | Visit1 | 1 | EYE | RETINAL | 0 | RNICK.NICKRADIO | | N
PA1(1) | Visit1 | 1 | EYE | RETINAL | 0 | LNICK.NICKRADIO | | N
PA1(1) | Visit1 | 1 | HH | PT | PT | 1 | THERAPYTEXT.THERAPYTEXT | | Name of a doctor
```

The following export file fragment is an example of a repeating form:

```
ABC(101) | CommonCRF | 1.000 | L_ConMeds | 291130301735008.000 | L_ConMeds | | 0.000 | CMHidden
J.CMHiddenJ | | test
```

The following export file fragment is an example of an association:

```
BBB(111) | CommonCRF | 1.000 | LAE1 | 319514859682043.000 | "BBB(111) |
CommonCRF | 1.000 | L_ConMeds | 319587403827450.000 "
```

# Exporting data in Oracle Clinical format

## Overview

The **Oracle Clinical** output option of the InForm Data Export utility outputs data from the InForm database in a format that you can upload to an Oracle Clinical database using the Oracle Clinical Upload application.

The Oracle Clinical output option of the InForm Data Export utility exports data in the following output file formats:

- **OA6 file**—Main output that contains rows whose ID/key changed since the date of the last run. If a path is not mapped, the InForm Data Export utility does not output the corresponding data. Use the OA6 file when you import data to an Oracle Clinical database.
- **DEL file**—Contains information about deleted records. You can reference this file to manually delete data from external databases.
- **NM file**—Contains the paths of unmapped elements.
- **log file**—Contains information about the export.

## Transferring data to the Oracle Clinical upload application

To set up data transfer to the Oracle Clinical Upload application:

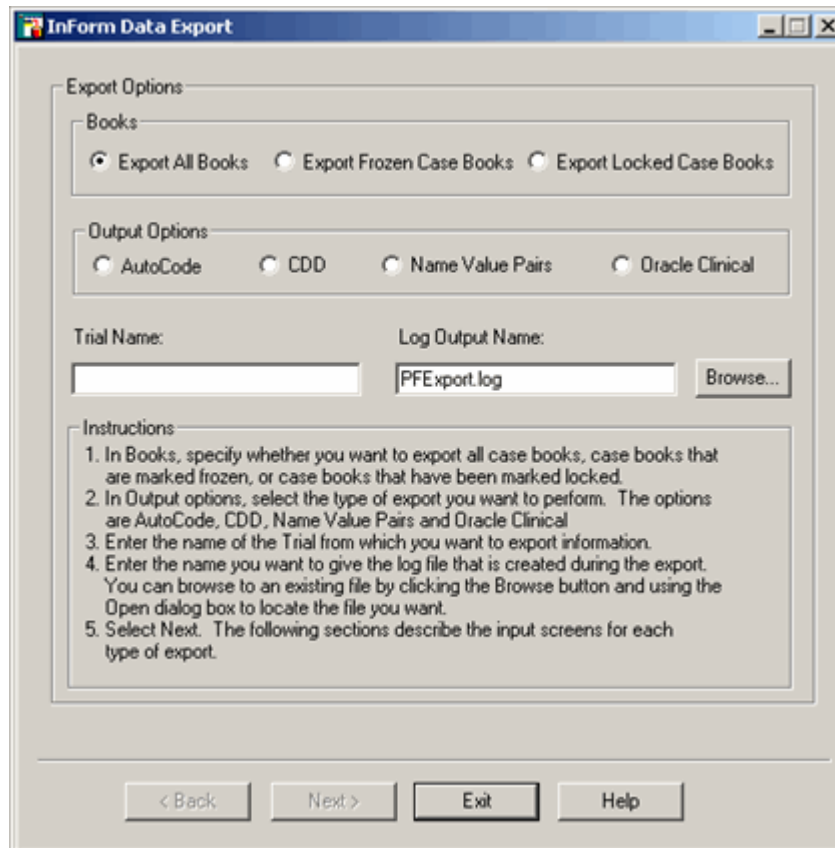
- 1 Use the InForm Architect application to generate an Oracle Clinical mapping definition for the trial, and save it in a location that you can access while running the MedML Installer utility.  
The mapping definition is a set of XML tags that specify how the controls in the trial map to Oracle Clinical database fields. For information about generating an Oracle Clinical mapping definition, see the *Setting Up a Trial with InForm Architect and MedML Guide*.
- 2 Load the mapping definition into the trial database using the MedML Installer utility to process the mapping XML file. For information about using the MedML Installer utility, see the MedML online help, or Using the MedML Installer utility.
- 3 Produce a fixed-length or character-separated flat file of the mapped data using the InForm Data Export utility.
- 4 Import the output file into an Oracle Clinical database by using the Oracle Clinical Upload application.

## Running the export in Oracle Clinical format

To run the Oracle Clinical data export:

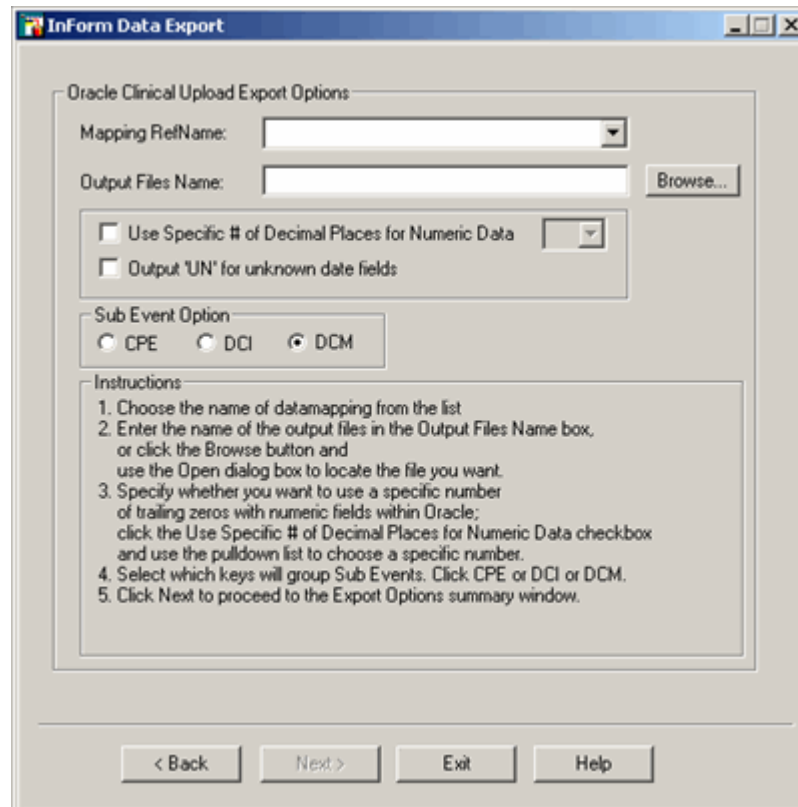
- 1 Click **Start > Programs > Oracle Health Science > InForm 4.6.5 > InForm Data Export utility**.

The InForm Data Export utility window appears.



- In the **Books** section, select which case books to export.
  - In the **Output Options** field, select **Oracle Clinical**.
  - Specify the **Trial Name** and the **Log Output Name**, or click **Browse** and locate the log output name.
- 2 Select **Next**.

The Oracle Clinical Upload Export Options window appears.



- 3 In the **Mapping RefName** drop-down list, select the RefName of the Oracle Clinical mapping definition that you created with the InForm Architect application.
  - In the **Output Files Name** field, specify the name of an output file. The InForm Data Export utility adds the suffixes for each file type it generates to the name that you specify.
  - Select the **Use Specific # of Decimal Places for Numeric Data** checkbox and use the drop-down list to specify the number of trailing zeros to use in numeric fields within the Oracle database.
  - Select the **Output UN for unknown data fields** checkbox to flag unknown data fields in your export.
  - Select a **SubEvent** option:
    - **CPE**—SubEvents are grouped by Clinically Planned Event.
    - **DCI**—SubEvents are grouped by Data Collection Instrument, a customer-defined grouping of Data Collection Modules.
    - **DCM**—SubEvents are grouped by Data Collection Module, a customer-defined grouping of data items.
- 4 Click **Next**.

The Run Date window appears.

**InForm Data Export**

Run Date

Incremental Export     Generate Data Comment

DateTime of Last Run (GMT): 4/6/2005 9:38:52 PM    <-- Use Suggested Time    Sugg. DateTime of Last Run (GMT): 4/6/2005 9:38:52 PM

Field Format:  Fixed     CSV, Separated by: [ ]

Output Version:  3.1.1     3.0    Study: [ ]

Deleted Items in ItemSets:  Output to Delete File     Export with Blank Values

[ Site Mapping ]

Number	Site Name
MCLEAN	MCLEAN
MGH	MGH
PF	PF
VA	VA

[ Investigator Mapping ]

Investigator	Site Name
	MGH
	VA
	MCLEAN
	PF

< Back    Next >    Exit    Help

5 In the **Run Date** section:

- Select **Incremental Export** to receive an export of only the data entered after a certain date/time.
  - If you selected **Incremental Export**, specify the DateTime after which any data entered is to be exported, or click **Use Suggested Time** to enter the time shown in the **Sugg. DateTime of Last Run (GMT)** box.
- Select **Generate Data Comment** to export comments as well as data.

The run date is also used to determine if a visit date, site number, patient number, or investigator number has changed since the last run date. If there has been a change, a delete record is produced for the appropriate item.

**Note:** By default, the values in the **DateTime of Last Run** and **Suggested DateTime of Last Run** fields are expressed in Greenwich Mean Time (GMT). If you enter a different date and time than the default or suggested values, you must be sure to use GMT.

6 In the **Field Format** section:

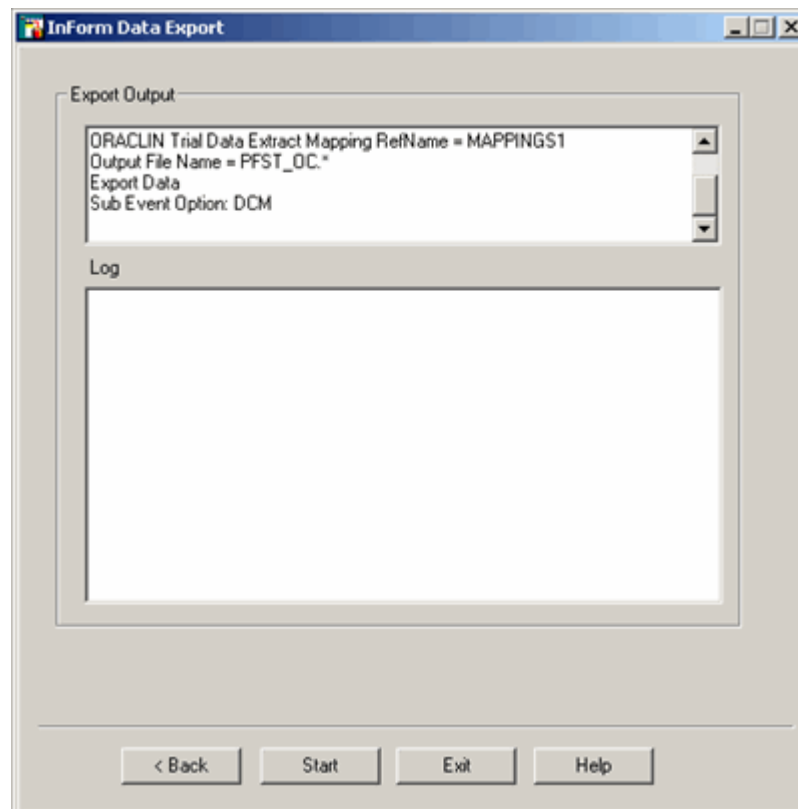
- Select **Fixed**
- or
- Select **Comma-separated value (CSV)**

- In the **Output Version** section, select the version of the Oracle Clinical Batch Upload tool to use.

**Note:** If you select version 3.1.1, you must type the name of the trial.

- In the **Deleted Items in ItemSets** section:
  - Select **Output to Delete File** to output the deleted itemset rows to the DEL file and to resequence the remaining itemset rows.
  - Select **Export with Blank Value** to keep the deleted itemset rows in the export file with blank values.
- In the **Site Mapping** section, map the site mnemonics and numbers by double-clicking the numbers to change them.
- In the **Investigator Mapping** section, map the site mnemonics and investigators by double-clicking the investigators to change them.
- Click **Next**.

The Export Output window appears with the data that you specified.



- Click **Start** to begin the export.

The InForm Data Export utility begins to create the OA6 file and display progress messages in the log area of the window. These messages are logged in the LOG file whose name you specified in the Oracle Clinical Upload Export Options window.

- When the export finishes, click **Exit**.

## Running the InForm Data Export utility from the command line

To run the InForm Data Export utility from the command line, use the following parameters.

**Note:** Oracle strongly recommends that you run the InForm Data Export utility through the PFConsole utility.

Parameter	Variable	Description
<b>PFConsole utility</b>		Starts the PFConsole utility.
<b>PFExport</b>		Starts the InForm Data Export utility.
-autorun		Runs the InForm Data Export utility in a command window.
-?		Displays the usage statement.
-help		Displays the online help for the InForm Data Export utility.
-Trial	trial name	The name of the trial from which to export data.
-outfile	output log file name	Indicates that the next parameter is the name of an output file.  Full pathname of an output file that contains the text of messages displayed by the InForm Data Export utility. Optional.
-cdd	ODBC DSN Name	Indicates that you are running the CDD Output tool.  ODBC DSN name of the CDD to use as the export target of the CDD Output tool is required. If you specify a DSN name, you must also provide <code>-user</code> and <code>-password</code> parameters.
-user	ODBC user name	Indicates that the next parameter is a user name.  User name of the user you set up for CDD access by the CDD Output tool; required if you specify a DSN name.
-password	ODBC password	Indicates that the next parameter is the user password.  Password of the CDD user you set up for CDD access by the CDD Output tool is required if you specify a DSN name.
-CreatDSN	OraConnStr	For CDD export only, creates a new DSN within the CDD database.



Parameter	Variable	Description
-RefName	CDDRefName	For CDD export only, drops the current user and creates a new schema that is defined by the RefName.
-TBSP	OraTabSpace	For CDD export only, the name of the tablespace in which to put the new schema when creating a new DSN within the CDD database.
-crfhelp		Indicates that you want to export CRFHelp.
-nvfile	output nv file name	Indicates that you are running the InForm Data Export utility for Name Value pairs.  Name of the export file that is created when you run the InForm Data Export utility for Name Value pairs is required if you specify the -nvfile flag.
-DelPrefix	string to prefix data marked as deleted	String that is added to a row of data in the export file to indicate that the data had been deleted from the InForm application prior to the export.
-Autocode		Indicates that you want to export for autocoding.
-REFNAME	targetset refname   ALL	One of the following: <ul style="list-style-type: none"> <li>The RefName of the predefined Autocode targetset.</li> <li>ALL, to export all predefined Autocode targetsets.</li> </ul>
-ACOUTFILE	output file name (*.xml)	Name of the XML to which you want to export.
-SEPARATE		Output each of the RefNames that are indicated above into separate files. If not specified, all RefNames are output to one file.
-NONCODEDONLY		Output only non-coded items.
-template	Oracle Clinical template file name	For export only to the Oracle Clinical database, required. Specify the path name of the template file to use.
-oraexport	DATA   SAVETEMPLA TE   CREATETEMP LATE	For Oracle Clinical export only, required. Specify either to export data, save the template as an XML file, or create the template from an XML file.
-numberdecimalplaces	# of decimal places	For Oracle Clinical export only, optional. Specify the number of trailing decimal places, between 0 and 10.
-SubEventOption	CPE   DCI   DCM	Determines which keys will group SubEvents.

Parameter	Variable	Description
-OutputUN		Outputs UN for unknown date fields, if present.
-sysadmin		Name of the System Administrator with rights to export CRF Help.  Indicates that the next parameter is the System Administrator ID. This ID is necessary to export CRFs.
-outputtype	ALL EPIC STANDARD	Indicates the type of export.
-customheader	text of header	For PDF format only, optional. Allows you to indicate the name of a custom header.
-template	template path name	For export only to the Oracle Clinical database, required. Specify the path name of the template file to use.
-oraexport	DATA SAVETEMPLATE CREATETEMPLATE	For export only to the Oracle Clinical database, required. Specify either to export data, save the template as an XML file, or create the template from an XML file.
-numberdecimalplaces	# of decimal places	For export only to the Oracle Clinical database, optional. Specify the number of trailing decimal places, between 0 and 10.
-RefName	defined CDD RefName	For CDD Export only, drops the current user and creates a new schema defined by the RefName.
-CreateDSN	Oracle connection string	For CDD Export only, creates a new DSN within the CDD database.
-TBSP	Oracle tablespace	For CDD Export only, the name of the tablespace in which to put the new schema.

## Example

The following is a sample syntax for running the InForm Data Export utility from the command line:

```
PFConsole
PFExport [-autorun] [-?] [-help] [-Trial trial name] [-outfile outfile name] [-
cdd ODBC DSN Name] [-user ODBC username] [-password ODBC password]
[-CreatDSN OraConnStr][-RefName CDDRefName][-TBSP OraTabSpace]
[-crfhelp]
[-nvfile nv file name [-DelPrefix string to prefix data marked as deleted]]
[-Autocode [-REFNAME targetset refname|ALL] -ACOUTFILE output file name
(*.xml)[-SEPARATE] [-NONCODEDONLY]]
[-Template Oracle Clinical Template file name
[-REFNAME targetset refname]
[-Ora Export DATA|SAVETEMPLATE|CREATETEMPLATE]
[-NumberDecimalPlaces # of decimal places]
[-SubEventOption CPE|DCI|DCM]
[-OutputUN]]
```

## Oracle Clinical fields

The following are the Oracle Clinical fields for Oracle Clinical data exports:

Oracle Clinical field	Description
Investigator	Investigator (FINO) number—Customer-defined number to uniquely identify an investigator.
Site	Site number—Customer-defined character value (mostly numbers, may have leading zeros) to uniquely identify the site.
Patient number	Patient identification—A numeric value that is assigned to a patient by the investigator. A patient number is unique across a protocol.
Document number	Customer-defined number to identify the document.
Clinical Planned Event (CPE) name	Customer-defined visit name—A character value that describes in detail the name of the visit as specified in the protocol timetable.
Subevent Number	A unique value that is assigned to a particular event (patient, visit and DCM) when the result date or time information does not match the previous date or time for the given event.
Visit Date	Date of visit—Format YYYYMMDD.
Visit Time	Time of visit—Format HHMMSS. (Depends on protocol. Leave blank if not used.) For lab data, if a result has time associated with the DCI Name, you must use LAB TIME.
DCI Name	Data collection instrument (DCI) Name—Customer-defined name to identify a grouping of data collection modules (DCMs). For example, lab data DCMs HEMATOLOGY_01 and CHEMISTRY_01 might be assigned to the DCI Name LAB.
DCM Name	Data collection module name—Customer-defined name to identify the grouping of data. For example, Red Blood Cell count, Hemoglobin, and Hematocrit might be assigned to the HEMATOLOGY_01 DCM.
DCM Subset Name	Customer-defined name to specify how data is divided into smaller subsets by data type.
Question Group Name	Customer-defined name to specify how different results are grouped together.
Question Name	Data point identifier (for example, lab test performed) that is translated into a standard variable name as provided by the customer.
Occurrence Number	Occurrence number that identifies the test, and that is used to associate repeating data, such as repeated labs. Default value should be 0 (zero). A change in occurrence number would be used to indicate a retest value.

Oracle Clinical field	Description
Repeating Sequence Number	Increment for each result starting at 1 (one). Can also be used in conjunction with the Occurrence number. (A common repeat sequence number indicates a relationship between values.)
Question Value	Answer to the question, or the test result. Can contain numeric or text values as needed.
Comment	Data comment text (optional); also used to indicate record deletions.
Qualifying Question Value	Customer-defined value.
Study	Trial name that is in the form CI_PROT (for example, 1008_32); no leading zeros on CI or PROT.

## Format of output files

### Format of OA6 file

This section of an OA6 file shows the fields that are exported according to the table in *Oracle Clinical fields* (on page 395). Each field below corresponds to a field that is listed, in order of presentation. The file is fixed format.

```

0001 001 1111 V1 0 20000617
DEMOGRAPHICS/DRUG SENSITIVITYDEMOGRAPHICS 01 DEMOG PATIENT INFORMATION 01
PTINTL 0 1 AAA
0001 001 1111 V1 0 20000617
ELIGIBILITY ELIGIBILITY 01 ELIGIB ELIGIBILITY 01
ELIG 0 1 Yes
0001 001 1111 V3/RANDOMIZATION 0 20000404
SUBJECT STATUS-SCREENING PATIENT STAT 02 STATUS1 PATIENT STATUS 02
PHASE 0 1 Screening
0001 001 1111 V3/RANDOMIZATION 0 20000404
SUBJECT STATUS-SCREENING PATIENT STAT 02 STATUS1 PATIENT STATUS 02
STATUS 0 1 Completed

```

### Format of DEL file

This section of a DEL file shows the records for an investigator key that was changed from 6666 to 5555. This delete file was exported with comma-separated values turned on, meaning that a delete record is generated for every Visit/Visit Date for this investigator's patients.

```

6666,002,1111,,V10,0,20000802,,,,,,,,0,1,Investigator key has changed from '6666'
to '5555',DELETE,
6666,002,1111,,V5,0,20011108,,,,,,,,0,1,Investigator key has changed from '6666'
to '5555',DELETE,

```

## CHAPTER 6

# InForm Performance Monitor utility

### In this chapter

Overview .....	398
Starting the InForm Performance Monitor utility.....	399
Capturing performance statistics .....	400
Performance Monitor output options .....	403
Managing the InForm Performance Monitor data.....	404
Examples of using the InForm Performance Monitor utility.....	405

## Overview

The InForm Performance Monitor utility runs on the desktop where an InForm application server is running. When a session of the InForm application is active, the InForm Performance Monitor utility listens for and captures InForm application server messages about specific types of InForm activities, and displays the messages in a window where you can arrange and save them. You can use this utility to:

- Provide information that helps with performance tuning during trial development and implementation.
- Capture performance data for troubleshooting.

**Note:** The InForm Performance Monitor utility is not intended for extended use in a production environment because it can impact server performance over time. Oracle suggests running the InForm Performance Monitor utility at 5-10 minute intervals.

# Starting the InForm Performance Monitor utility

To start the InForm Performance Monitor utility:

- 1 Select **Start > Programs > Oracle Health Science > InForm 4.6.5 > InForm Performance Monitor**.

The Performance Monitor window opens. If one or more InForm servers on the machine are running, the InForm Performance Monitor utility begins recording InForm application server messages.

TrialID	RequestID	Time(ms)	Time	Tag	Message
0	0	16	3/20/01 9:56:54 PM	DoExecQuery	select * from PF_SiteFilter where VectorID=6 AND
0	0	15	3/20/01 9:56:59 PM	DoExecQuery	select * from PF_SiteFilter where VectorID=8200 A
0	0	16	3/20/01 9:57:04 PM	DoExecQuery	select * from PF_SyncQueue where 1=1
0	0	15	3/20/01 9:57:40 PM	DoExecQuery	select * from DCV_SyncVector where VectorID =
0	0	15	3/20/01 9:57:50 PM	DoExecQuery	select * from PF_SiteFilter where VectorID=8200 A
0	0	15	3/20/01 9:58:10 PM	DoExecQuery	select * from PF_SiteFilter where VectorID=8200 A
0	0	15	3/20/01 9:58:31 PM	DoExecQuery	select * from PF_SiteFilter where VectorID=8200 A
0	0	16	3/20/01 9:59:21 PM	DoExecQuery	select * from PF_SiteFilter where VectorID=8200 A
0	0	16	3/20/01 10:00:02 PM	DoExecQuery	select * from PF_SiteFilter where VectorID=8200 A
0	0	16	3/20/01 10:00:28 PM	DoExecQuery	select * from PF_SiteFilter where VectorID=8200 A
0	0	16	3/20/01 10:00:38 PM	DoExecQuery	SELECT VectorId, State FROM Dcv_SyncVector
0	0	15	3/20/01 10:00:38 PM	DoExecQuery	select * from PF_SiteFilter where VectorID=8200 A
0	0	15	3/20/01 10:01:03 PM	DoExecQuery	select * from PF_SiteFilter where VectorID=8200 A
0	0	15	3/20/01 10:02:04 PM	DoExecQuery	select * from PF_SiteFilter where VectorID=8200 A
0	0	16	3/20/01 10:02:09 PM	DoExecQuery	SELECT VectorId, State FROM Dcv_SyncVector
0	0	15	3/20/01 10:02:10 PM	DoExecQuery	select * from PF_SyncQueue where 1=1
0	0	15	3/20/01 10:02:10 PM	DoExecQuery	select * from DCV_SyncVector where VectorID =
0	0	15	3/20/01 10:03:21 PM	DoExecQuery	select * from PF_SiteFilter where VectorID=8200 A
0	0	16	3/20/01 10:03:31 PM	DoExecQuery	select * from PF_SiteFilter where VectorID=8200 A
0	0	15	3/20/01 10:03:57 PM	DoExecQuery	select * from PF_SiteFilter where VectorID=8200 A
0	0	16	3/20/01 10:04:02 PM	DoExecQuery	select * from DCV_SyncVector where VectorID =
0	0	16	3/20/01 10:04:12 PM	DoExecQuery	select * from PF_SiteFilter where VectorID=8200 A
0	0	15	3/20/01 10:04:17 PM	DoExecQuery	select * from PF_SiteFilter where VectorID=8200 A
0	0	16	3/20/01 10:04:22 PM	DoExecQuery	select * from PF_SiteFilter where VectorID=8200 A
0	0	16	3/20/01 10:04:27 PM	DoExecQuery	select * from PF_SiteFilter where VectorID=8200 A
0	0	16	3/20/01 10:04:42 PM	DoExecQuery	select * from PF_SiteFilter where VectorID=8200 A

**Note:** The InForm Performance Monitor utility is used by Oracle staff to assess and enhance the performance of a trial, and evaluate server activities by quantifying the amount of time each activity requires. Although a brief description of its messages is detailed in *Capturing performance statistics* (on page 400), the use and interpretation of the InForm Performance Monitor utility is at the sole discretion of Oracle staff.

## Capturing performance statistics

For each activity in a session of the InForm application, the InForm Performance Monitor utility captures the following data and displays it in the Performance Monitor window:

- **Trial ID**—Specifies the trial that is associated with an InForm Performance Monitor utility message. If a message is not trial-specific, it contains a value of 0.
- **Request ID**—Contains a numeric value. The InForm Performance Monitor utility messages with the same, non-zero RequestID value are associated with some common activity. For example, in the figure below, the messages that have a RequestID value of 25016952 are all associated with submitting a CRF (last line of figure).

TrialID	RequestID	Time(ms)	Time	Tag	Message
0	0	20	10/31/01 8:30:00 PM 663...	CPFUData: Query DC...	select * from DCV_ThingStiData where TypeName = BAF5075C-1C29-11d4-A126-000B709DF4E
0	0	7	10/31/01 8:30:00 PM 723...	UDACore_InsertRowData	Insert into PF_Session (CREATIONDATE TIME, BROWSECOLOR, CREATIONSECONDS, USERID)
0	0	13	10/31/01 8:30:01 PM 193...	CPFUData: Query DC...	SELECT I FROM DCV_ClinicalData WHERE (KeyID = 8910 AND FormsetID = 8278 AND FormID =
0	0	18	10/31/01 8:30:03 PM 467...	CPFUData: Query DC...	select * from DCV_Candidate where SiteID = 6126
0	0	10	10/31/01 8:30:04 PM 538...	CPFUData: Query DC...	SELECT MDS.MESSAGEDATA, MS.STATUS, MI.MESSAGEID, MI.TRANSPORTID FROM PF_SY
0	0	10	10/31/01 8:30:04 PM 548...	CPFUData: Query DC...	select * from DCV_SyncVector where VectorID = 17344
0	0	10	10/31/01 8:30:04 PM 568...	CPFUData: Query FF...	select * from PF_SiteFilter where VectorID=17344 AND SiteID>0 AND RevisionNumber=(SELECT M
0	0	6	10/31/01 8:30:06 PM 581...	CPFUData: Query FF...	SELECT * FROM (SELECT ROWNUM AS NUM, I FROM (SELECT DCV_EnrolledPatient.Pa
0	0	5	10/31/01 8:30:09 PM 616...	CPFUData: Query DC...	SELECT NeedsRecalcState FROM PF_SubjectVEChapterPage WHERE subjectchapterid =
0	0	45	10/31/01 8:30:09 PM 646...	CPFUData: Query DC...	select * from DCV_QueryAuditComment where KeyID=8910 AND FormsetID=8260 AND FormsetInde
0	0	49	10/31/01 8:30:09 PM 696...	CPFUData: Query DC...	SELECT QueryID, QueryRevisionNumber, ContextID, QueryType, QueryState, QueryGroup, I
0	0	6	10/31/01 8:30:29 PM 294...	CPFUData: Query FF...	select * from PF_TransactionHistory where InternalGUID = '9052F1E-CE3C-11D5-AC3F-00034770
0	0	6	10/31/01 8:30:29 PM 304...	CPFUData: Query FF...	select * from PF_SubjectVEChapter where (SubjectChapterRevisionNumber = (SELECT MAX(Subjec
0	0	8	10/31/01 8:30:29 PM 314...	CPFUData: Query FF...	select * from PF_SubjectVEChapterPage where (PageID = 7198) AND SubjectChapterID IN (SELEC
0	0	8	10/31/01 8:30:29 PM 324...	CPFUData: Query FF...	select * from PF_ItemContext where (SubjectKeyID = 8910 AND ChapterID = 8260) AND PageID = 71
0	0	5	10/31/01 8:30:29 PM 324...	CPFUData: Query FF...	select * from PF_SubjectVEChapter where SubjectKeyID = 8910 AND ChapterID = 8260 AND Chap
0	0	39	10/31/01 8:30:29 PM 394...	UDACore_InsertRowData	Insert into PF_SubjectVEChapter (DELETED, SUBJECTCHAPTERREVISIONNUMBER, ACTUALDA
0	0	71	10/31/01 8:30:29 PM 444...	UDACore_InsertRowData	Insert into PF_SubjectVEChapterPage (READYFORSDV, HASCOMMENTSSTATE, NEEDSUNSIGNA
0	0	108	10/31/01 8:30:29 PM 564...	UDACore_InsertRowData	Insert into PF_ItemContext (CHAPTERINDEX, SUBJECTKEYTYPE, CONTEXTID, CHAPTERID, PAI
0	0	24	10/31/01 8:30:29 PM 714...	UDACore_InsertRowData	Insert into PF_ControlData (PARENTCONTROLVALUEID, CONTROLID, CONTROLVALUEID, INVA
0	0	29	10/31/01 8:30:30 PM 983...	CPFUData: Query Lj...	select * from L_ImdDV where PatientID=8910 and VisitID=8260 and VisitIndex=1.000 and ItemSetIn
0	0	44	10/31/01 8:30:31 PM 33 ms	CPFUData: Query DC...	SELECT * FROM PF_SVECPSTATEHISTORY sgrh WHERE sgrh.SubjectChapterID IN (select SU
0	0	24	10/31/01 8:30:31 PM 83 ms	CPFUData: Query DC...	SELECT ControlID FROM PF_ItemContext, PF_ControlData t WHERE t.ContextID = PF_ItemConte
0	0	46	10/31/01 8:30:31 PM 123...	CPFUData: Query DC...	SELECT COUNT(A.ITEMID) AS ITEMDCOUNT, JC.SDVSTATE FROM (SELECT CP.PAGEID, CP.SL
0	0	7	10/31/01 8:30:31 PM 183...	CPFUData: Query DC...	select * from DCV_QueryAuditComment where KeyID=8910 AND FormsetID=8260 AND FormsetInde
0	0	28	10/31/01 8:30:31 PM 193...	CPFUData: Query DC...	SELECT QueryID, QueryRevisionNumber, ContextID, QueryType, QueryState, QueryGroup, I
0	0	78	10/31/01 8:30:31 PM 274...	CPFUData: Query DC...	SELECT COUNT(IPF_Query.QueryID) AS QueryCount FROM PF_Query, PF_ItemContext WHERE
0	0	6	10/31/01 8:30:31 PM 304...	CPFUData: Query FF...	select * from PF_SubjectVEChapter where (SubjectChapterRevisionNumber = (SELECT MAX(Subjec
0	0	33	10/31/01 8:30:31 PM 514...	CPFUData: Query FF...	select * from L_ImdDV where PatientID=8910 and VisitID=8260 and VisitIndex=1.000 and ItemSetIn
0	0	41	10/31/01 8:30:31 PM 564...	CPFUData: Query Pabi...	select * from Patient where PatientID=8910
0	0	29	10/31/01 8:30:31 PM 614...	CPFUData: Query Pabi...	select * from Patient where PatientID=8910
0	0	8	10/31/01 8:30:31 PM 654...	CPFUData: Query FF...	select * from PF_SubjectVEChapterPage where SubjectChapterID = 21375 AND PageID = 7198
0	0	14	10/31/01 8:30:31 PM 684...	UDACore_InsertRowData	Insert into PF_RevisionHistory (TRANSACTIONID, REASON, KEYID, USERID, HISTORICALORDEF
0	0	30	10/31/01 8:30:31 PM 694...	UDACore_InsertRowData	Insert into PF_SVECPStateHistory (PAGEHISTORYID, PAGEHISTORYREVISIONNUMBER, STATE
0	0	60	10/31/01 8:30:31 PM 844...	CPFUData: Query DC...	SELECT I FROM DCV_ClinicalData WHERE (KeyID = 8910 AND FormsetID = 8260 AND FormID =
0	0	7	10/31/01 8:30:31 PM 915...	CPFUData: Query DC...	select * from DCV_QueryAuditComment where KeyID=8910 AND FormsetID=8260 AND FormsetInde
0	0	7	10/31/01 8:30:31 PM 925...	CPFUData: Query DC...	SELECT QueryID, QueryRevisionNumber, ContextID, QueryType, QueryState, QueryGroup, I
0	0	10	10/31/01 8:30:34 PM 578...	CPFUData: Query DC...	select * from DCV_SyncVector where VectorID = 17344
0	0	10	10/31/01 8:30:34 PM 588...	CPFUData: Query DC...	select * from DCV_SyncVector where VectorID = 6
pf30demo	25016952	67	10/31/01 8:30:29 PM 224...	Run Calcs on Form	
pf30demo	25016952	6	10/31/01 8:30:29 PM 764...	Run Rules on Form	
pf30demo	0	2292	10/31/01 8:30:29 PM 754...	CRF Submit CDD	pf30democdd
pf30demo	0	389	10/31/01 8:30:31 PM 163...	CRF Submit CDD	pf30democdd2
pf30demo	25016952	3660	10/31/01 8:30:29 PM 83 ms	Submit CRF	

- **Time (ms)**—Elapsed time in milliseconds from the user-issued request to the server response.
- **Time of transaction**—The time specified is GMT.
- **Tag**—Text string that describes the activity that is associated with the message.
- **Message**—Text string that provides detailed information that relates to the activity.

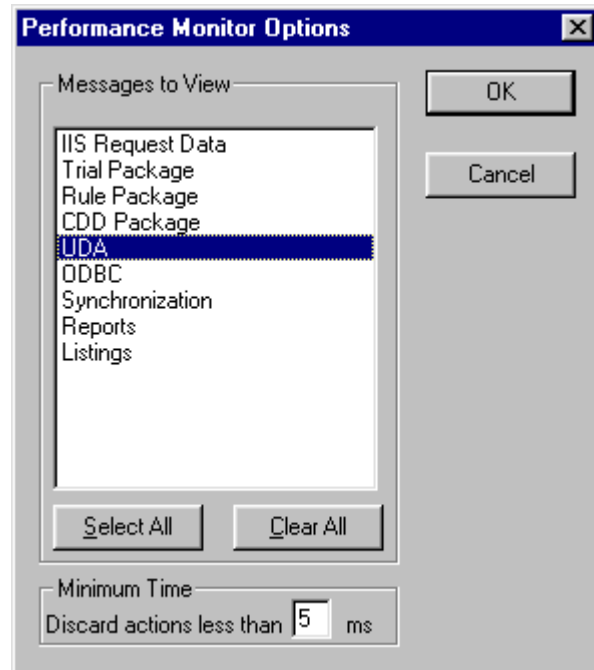
## Viewing messages from specific subsystems

To choose message filters:

- 1 Select **View > Options**.



The Performance Monitor Options dialog appears.



- In the **Messages to View** section, select one of the following filters to receive messages from only the specified subsystems:

Filter	Subsystem
IIS Request Data	The InForm ISAPI.
Trial package	The InForm trial MTS package, by InForm application server.
Rule package	The InForm rule package, by InForm application server.
CDD package	The InForm CDD package, by InForm application server.
UDA	The InForm database.
ODBC	The InForm ODBC.
Synchronization	The InForm Unplugged application.
Reports	The InForm report.
Listing	The InForm listing.
InForm service	The InForm service.
PFIimport	The InForm Data Import utility.

To select multiple subsystems, hold down the **Ctrl** key while you select.

- In the **Discard actions less than** field in the **Minimum Time** section, specify the threshold above which to record actions (in milliseconds).
- Click **OK**.

## Selecting InForm servers

To select an InForm application server from which to capture information:

- 1 Select **View > Servers**.

The InForm Server Selection List dialog appears.



- 2 Select the server from which to capture messages. To select multiple servers, hold down the **Ctrl** key while you select.
- 3 Click **OK**.

## Performance Monitor output options

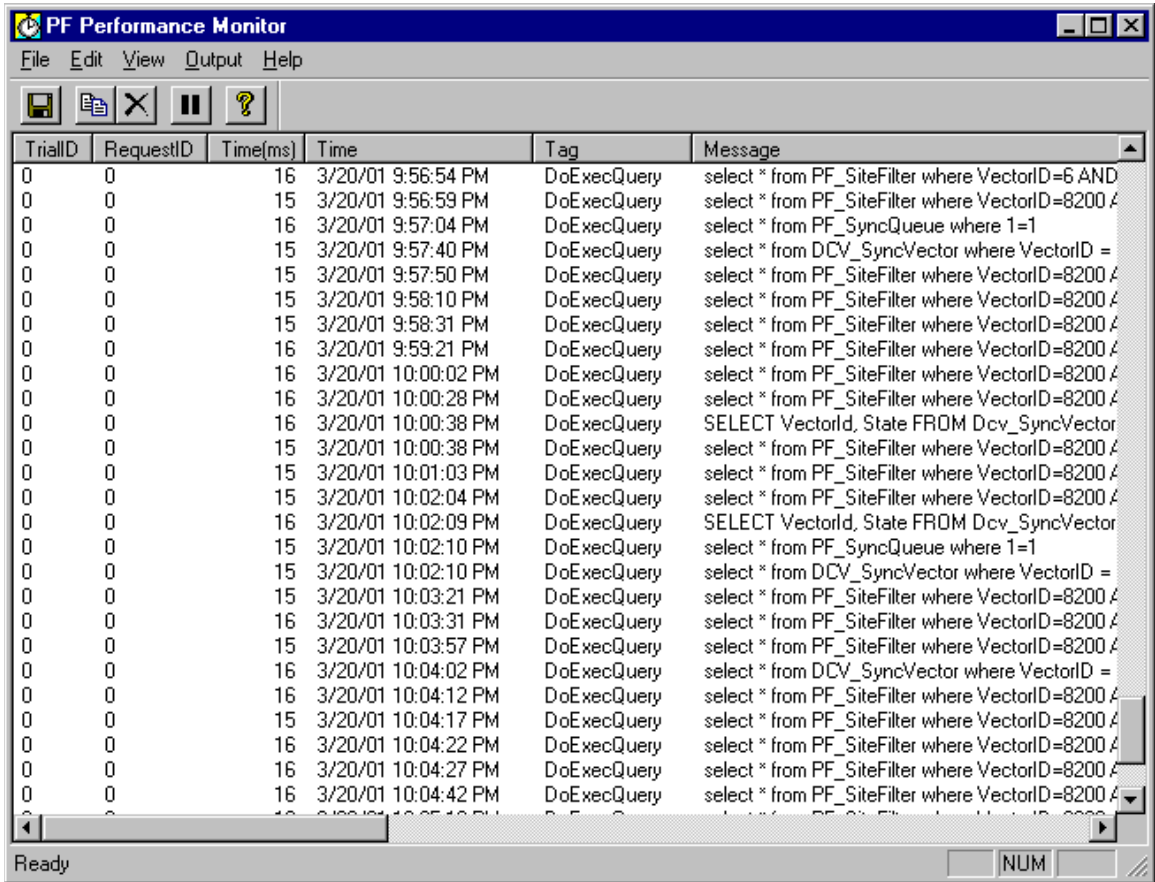
**Note:** Before you select an output option, select the statistics to capture as described in *Capturing performance statistics* (on page 400).

Select any of the following output options for the InForm Performance Monitor utility messages:

Output option	Description	Action
View output online	Displays messages in the Performance Monitor window. (Default.)	Select <b>Output &gt; Output to Window</b>
Stream output to a file	Sends messages to a file instead of (or in addition to) displaying them in the Performance Monitor window.	Select <b>Output &gt; Stream to File</b>
Save a performance log	Saves the performance log as comma-separated text to a specified file.	Select <b>File &gt; Save As</b>

## Managing the InForm Performance Monitor data

To change the display in the Performance Monitor window:



Task	Action
Sort a column	Click the column heading bar.
Select a single message	Click the message.
Select multiple contiguous messages	Hold down the <b>Shift</b> key while clicking the messages.
Select multiple noncontiguous messages	Hold down the <b>Ctrl</b> key while clicking the messages.
Select all messages	Select <b>Edit &gt; Select All</b> .
Copy selected messages	Select <b>Edit &gt; Copy</b> , or click the <b>Copy</b> button in the toolbar.
Delete selected messages	Select <b>Edit &gt; Delete</b> , or click the <b>Delete</b> button in the toolbar.
Clear all messages	Select <b>Edit &gt; Reset</b> .

# Examples of using the InForm Performance Monitor utility

## Testing rule script efficiency

After you develop a set of rules for a trial and add patient data to your test database, you can run the InForm Performance Monitor utility to:

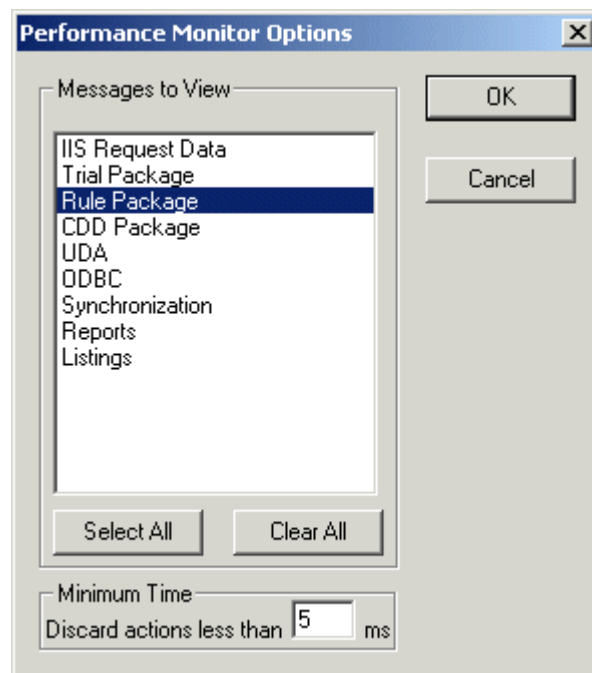
- Verify that rules are firing when expected, and that they are running against the expected rule contexts.
- Check for unusually long rule execution times.
- Determine how much of a transaction submit time is occupied by rule processing.

## Capturing rule processing data

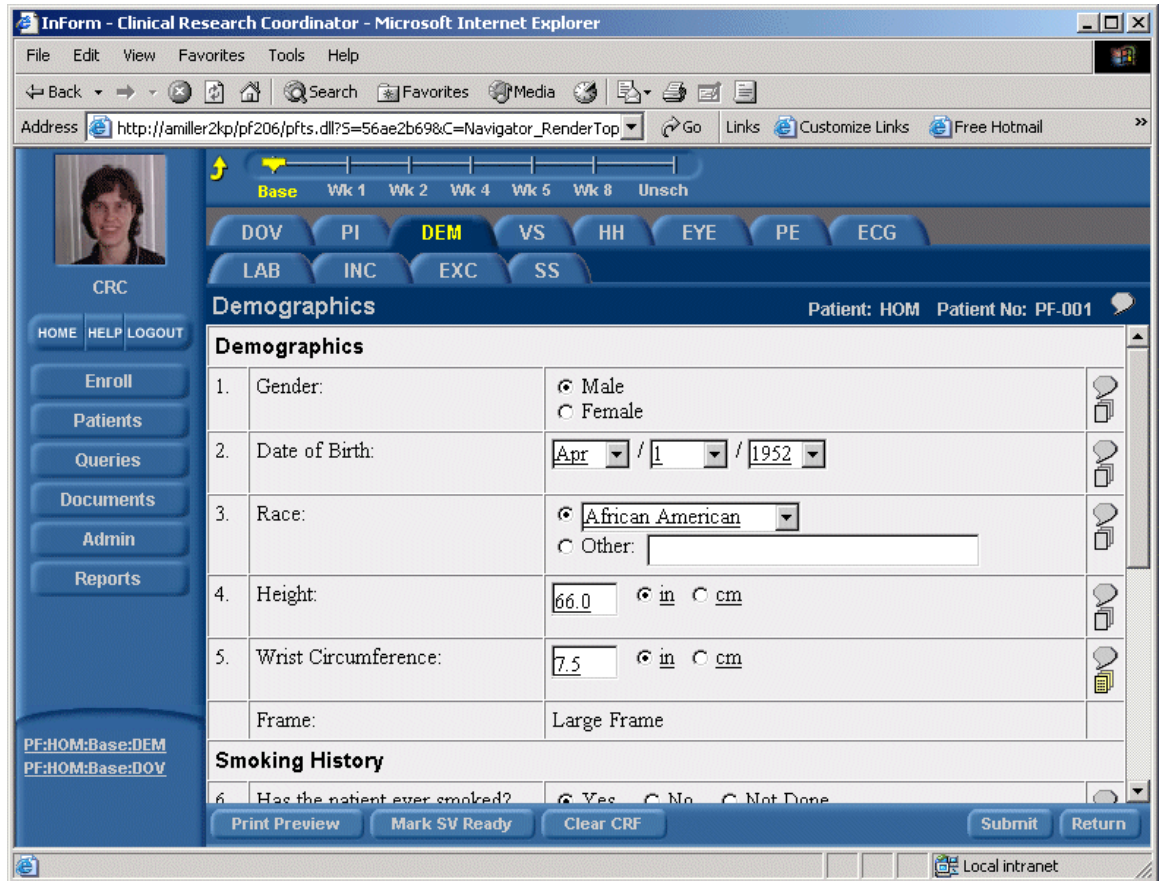
To capture rule processing data:

- 1 In the Performance Monitor window, select **View > Options**.

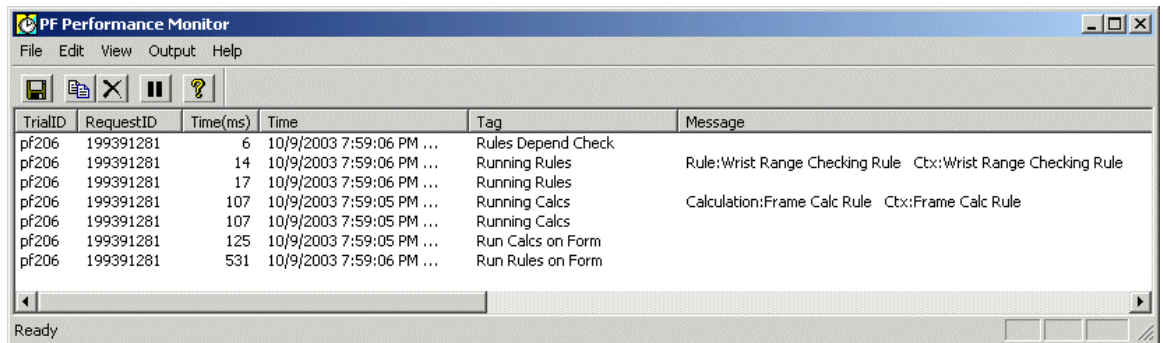
The Performance Monitor Options dialog appears.



- 2 Clear all filters except **Rule Package** to set the message filter to capture rule data.
- 3 Start the InForm trial and display the DEM form in the Baseline visit for one of the test patients.



- 4 Type or edit the value for the **Wrist Circumference** item, but do not click **Submit**.
- 5 In the Performance Monitor window, select **Edit > Reset** to clear the display.
- 6 In the InForm application, click **Submit**.
- 7 Return to the Performance Monitor window and check the messages.



The messages indicate that:

- The **Wrist Range Checking** and the **Frame Calc** rules fired as expected.
- The contexts for the rules were present.

The InForm Performance Monitor utility records the times that are required to check dependencies and to run the rules and calculations, as well as the total time to process rules on the form.

**Note:** To sort the messages by request times, click the **Time (ms)** column header. When you are gathering data on multiple rules, sorting by time can highlight rules that require unusually large amounts of time to process.

## Reviewing SQL query performance

If response times have degraded and your trial database accumulates data, you can use the InForm Performance Monitor utility to isolate long-running SQL queries. If you find long-running SQL queries, perform any of the following:

- Distribute tablespaces differently.
- Improve indexing on certain database tables.
- Run update statistics on the database.

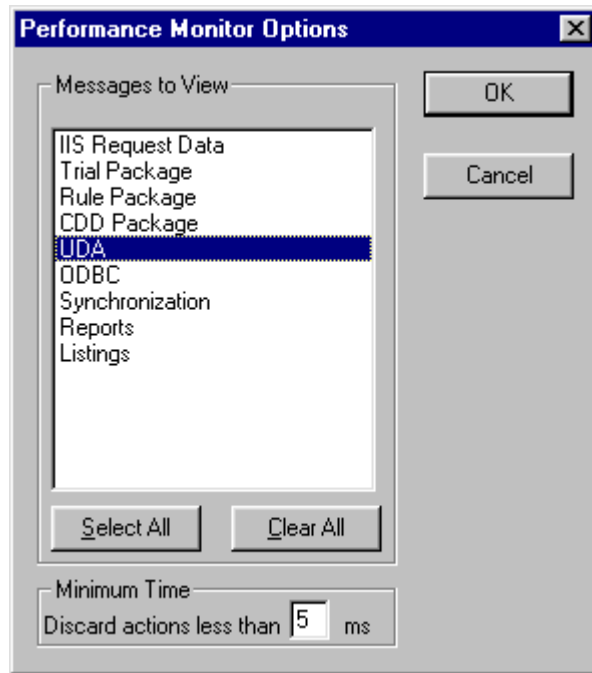
**Note:** The InForm Performance Monitor utility is not intended for extended use in a production environment because it can impact server performance over time. Oracle suggests running the InForm Performance Monitor utility at 5-10 minute intervals.

## Capturing SQL query performance data

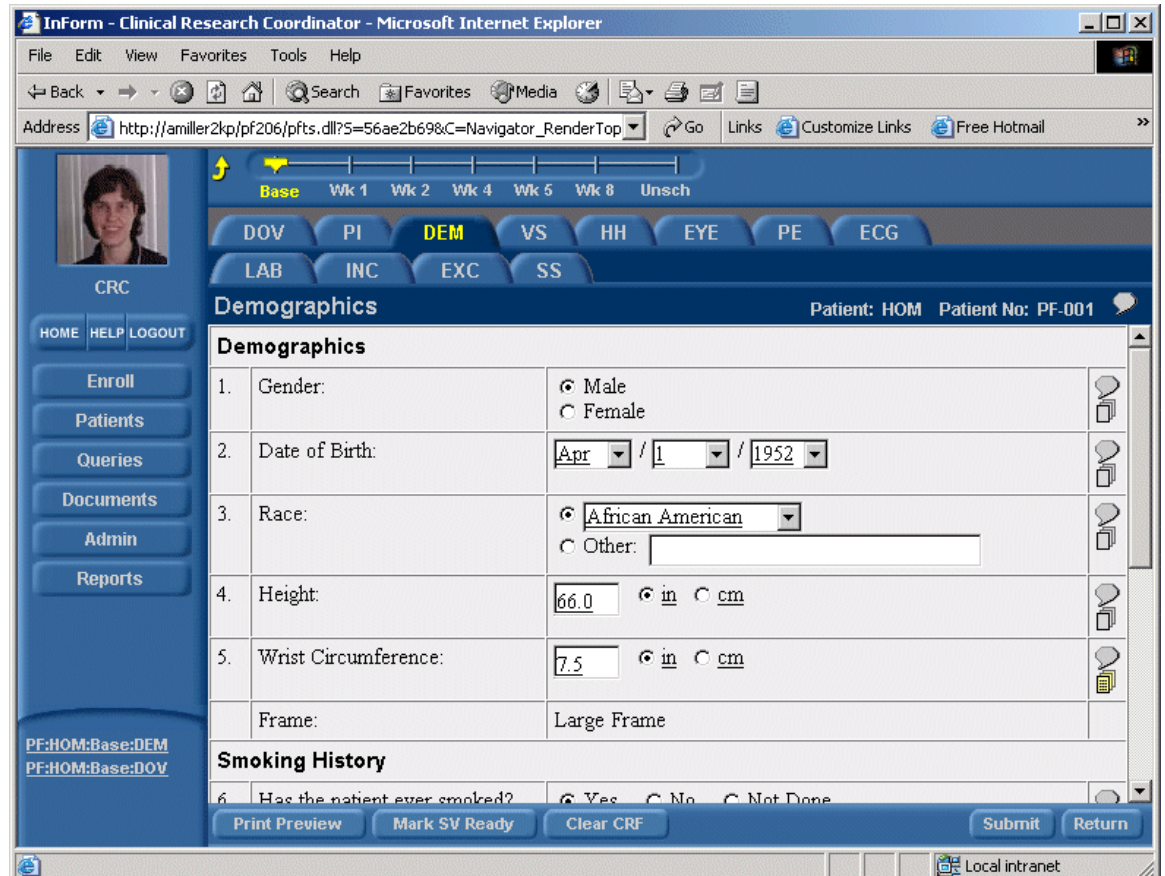
To capture SQL query performance data:

- 1 From the Performance Monitor window, select **View > Options**,

The Performance Monitor Options dialog appears.



- 2 Clear all filters except **UDA** to set the message filter to capture rule data.
- 3 Start the InForm trial and display the DEM form in the Baseline visit for one of the test patients.





- 4 Type or edit the value for the **Wrist Circumference** item, but do not click **Submit**.
- 5 In the Performance Monitor window, select **Edit > Reset** to clear the display.
- 6 Select **Output > Stream to File** to instruct the InForm Performance Monitor utility to stream the messages to a file, and specify the file name in which to store the messages.
- 7 In the InForm application, click **Submit**.
- 8 Return to the Performance Monitor window and check the messages.

TrialID	RequestID	Time(ms)	Time	Tag	Message
0	0	630	10/9/2003 8:38:20 PM ...	CPFUDataTable::Query P...	select * from PF_ExecutionPlanQueue where EPInstanceID = (SELECT N...
0	0	542	10/9/2003 8:38:20 PM ...	CPFUDataTable::Query P...	select * from PF_ExecutionPlanQueue where EPInstanceID = (SELECT N...
0	0	395	10/9/2003 8:38:20 PM ...	CPFUDataTable::Query P...	select * from PF_ExecutionPlanQueue where EPInstanceID = (SELECT N...
0	0	205	10/9/2003 8:38:22 PM ...	CPFUDataQuery::Execute	SELECT * FROM ( SELECT COUNT(*) AS SvedCount FROM DCV_ITEM...
0	0	117	10/9/2003 8:38:24 PM ...	CPFUDataQuery::Execute	SELECT QueryID, QueryRevisionNumber, ContextID, QueryType, Quer...
0	0	57	10/9/2003 8:38:20 PM ...	CPFUDataTable::Query P...	select * from PF_ExecutionPlanQueue where EPInstanceID = (SELECT N...
0	0	36	10/9/2003 8:38:25 PM ...	CPFUDataTable::NativeQuery	UPDATE PF_SubjectVEChapterPage SET NeedsRecalcState = 1 WHEREI...
0	0	27	10/9/2003 8:38:22 PM ...	CPFUDataQuery::Execute	SELECT ControlId FROM PF_ItemContext, PF_ControlData t WHERE t...
0	0	25	10/9/2003 8:38:25 PM ...	CPFUDataQuery::Execute	SELECT QueryID, QueryRevisionNumber, ContextID, QueryType, Quer...
0	0	24	10/9/2003 8:38:22 PM ...	UDACore_InsertRowData	Insert into PF_RevisionHistory (TRANSACTIONID, REASON, KEYID, USE...
0	0	18	10/9/2003 8:38:25 PM ...	CPFUDataTable::Query D...	select * from DCV_QueryAuditComment where KeyID=9489 AND Forms...
0	0	17	10/9/2003 8:38:21 PM ...	CPFUDataQuery::Execute	SELECT QueryID, QueryRevisionNumber, ContextID, QueryType, Quer...
0	0	16	10/9/2003 8:38:25 PM ...	UDACore_InsertRowData	Insert into PF_Query (QUERYSTATE, QUERYREVISIONNUMBER, CONFLI...
0	0	16	10/9/2003 8:38:25 PM ...	CPFUDataQuery::Execute	SELECT * FROM DCV_ClinicalData WHERE (KeyID = 9489 AND For...
0	0	13	10/9/2003 8:38:22 PM ...	CPFUDataQuery::Execute	SELECT * FROM PF_SVECPSTATEHISTORY sgnh WHERE sgnh.SubjectC...
0	0	13	10/9/2003 8:38:22 PM ...	CPFUDataTable::Query P...	select * from PF_TransactionHistory where InternalGUID = '{3F6AFFD...
0	0	12	10/9/2003 8:38:21 PM ...	CPFUDataQuery::Execute	SELECT * FROM DCV_ClinicalData WHERE (KeyID = 9489 AND For...
0	0	12	10/9/2003 8:38:25 PM ...	CPFUDataTable::Query P...	select * from PF_SubjectVEChapterPage where SubjectChapterID = 95...
0	0	11	10/9/2003 8:38:25 PM ...	UDACore_InsertRowData	Insert into PF_SVECPStateHistory (PAGEHISTORYID, PAGEHISTORYREV...
0	0	11	10/9/2003 8:38:22 PM ...	UDACore_InsertRowData	Insert into PF_TransactionHistory (TRANSACTIONID, REVISIONNUM, IMPR...

If requested to do so by Oracle support, send them the saved message file.

**Note:** To filter the stream of messages by the time required to execute the request, use the **Minimum Time** section of the Performance Monitor Options dialog. The InForm Performance Monitor utility displays only messages for requests that require more than the specified number of milliseconds to execute.

To determine a normal request time, enter 0 in the **Discard actions less than** field in the **Minimum Time** section, then compare the data in the **Time** field on the Performance Monitor window for each captured message.



## CHAPTER 7

# InForm Report Folder Maintenance utility

### In this chapter

Overview .....	412
Folder structure for multiple trials or sponsors .....	413
Setting up a folder structure for multiple trials or sponsors .....	418
Copying report folders .....	421

## Overview

The InForm Report Folder Maintenance utility is a Windows application that can do the following:

- Copy reporting folders and their contents to target folders.
- Define and create folder structures when using a single reporting server for
  - multiple trials
  - multiple sponsors, or
  - multiple trials within multiple sponsors.
- Automatically update any links to drill-down reports in InForm standard reports that were copied to a target location.
- Allow you to associate a copied report with a new reporting package at the target location.

Only users with System Administrator privileges can run the InForm Report Folder Maintenance utility.

**Note:** This includes saved reports as well as standard report definitions.

You install the InForm Report Folder Maintenance utility on the reporting (Cognos 10 Business Intelligence) server as part of the Reporting and Analysis installation and configuration. You can find it at this path:

`\c8\bin\PFMTRSetupUtil.exe`

The InForm Report Folder Maintenance utility copies only reports and report definitions. It does not copy the folders that contain links to legacy ASP reports, the InForm Trial Management package, or any published trial-specific clinical package. For more information about the association between reports and report packages, see *Report package association* (on page 419).

**Note:** You must complete the instructions in the *Configuring a Trial for Reporting* section in the *Installation and Configuration Guide* before performing the procedures in this chapter.

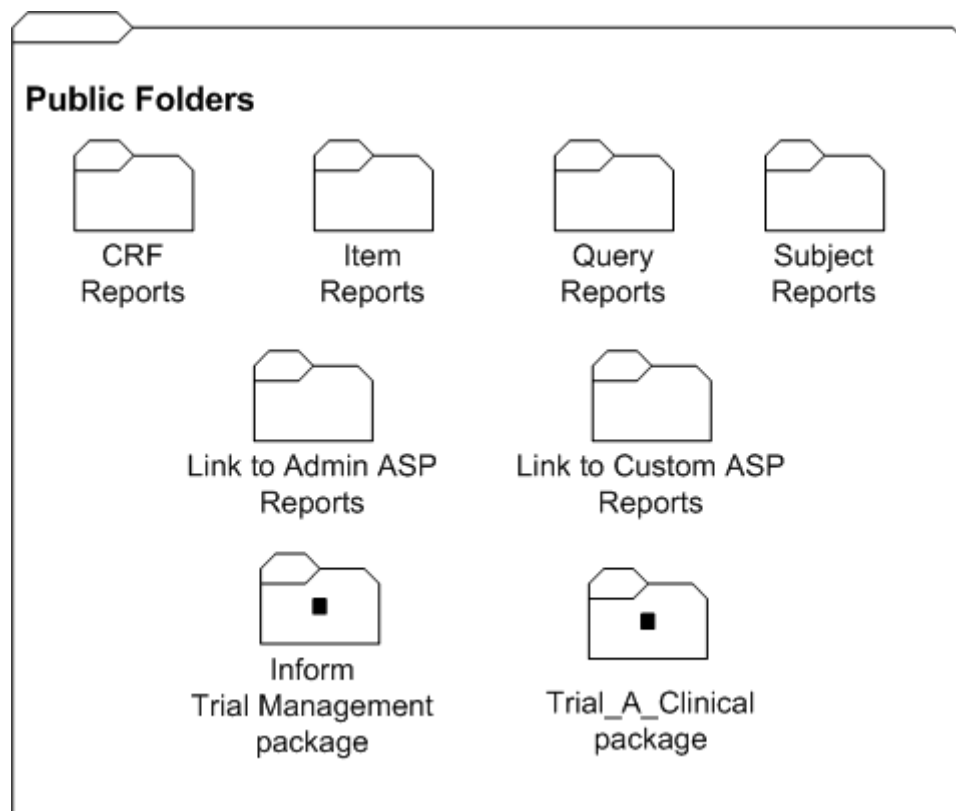
# Folder structure for multiple trials or sponsors

## Setting up the initial folder structure

To set up the initial folder structure:

- Import the standard reports archive. This archive includes all standard reports, as well as the InForm Trial Management package.
- Publish a trial-specific clinical package for clinical reports.

After you complete these steps, the InForm Report Folder Maintenance utility creates folders on the reporting server for the standard reports, legacy ASP reports, and the reporting packages. These folders appear in the **Public Folders** tab of the Reporting and Analysis portal. The following illustration shows the folders that appear after the initial setup of a single trial.



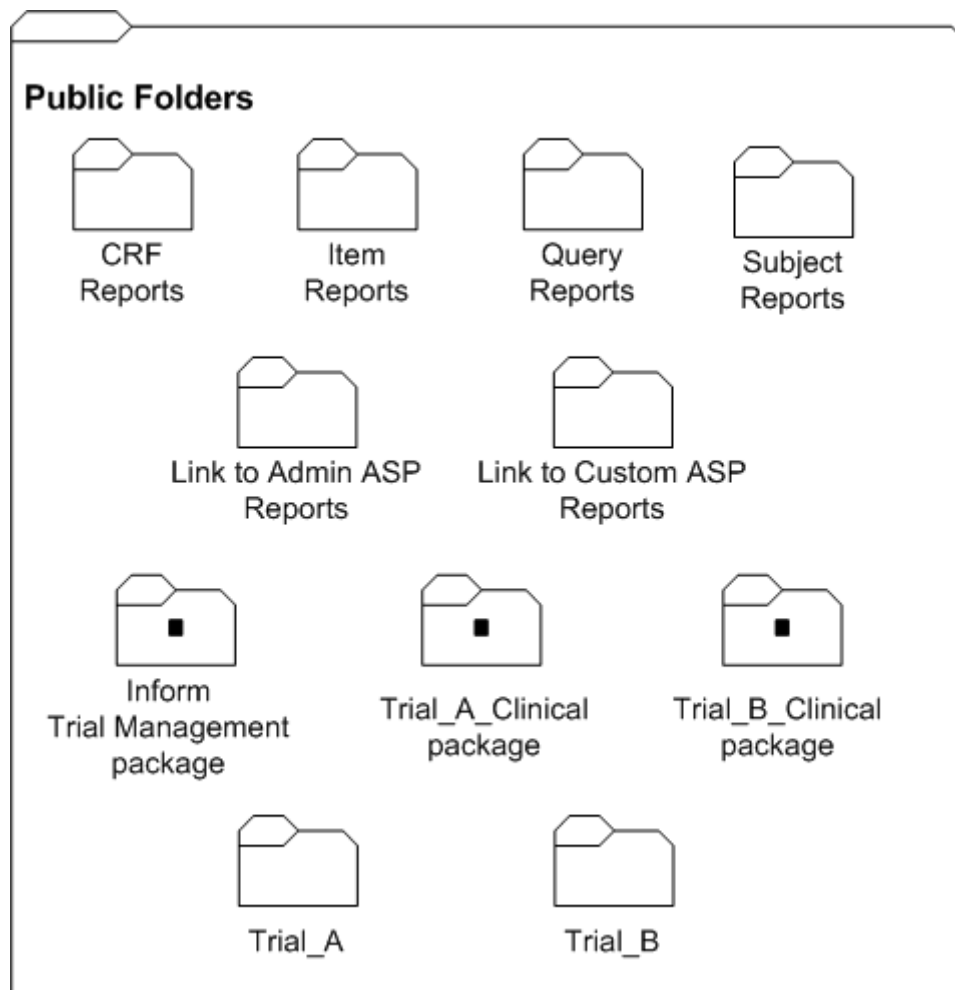
## Folder structure for multiple trials

To set up several trials for a single sponsor, consider creating a separate folder for each trial. PFRInit does this automatically.

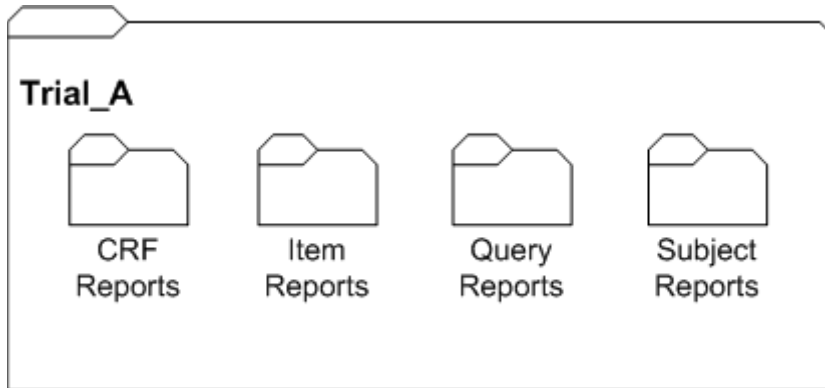
The following illustration shows how the **Public Folders** might look in the Reporting and Analysis portal with folders for two trials—Trial A and Trial B.

Note that this configuration uses three different packages:

- InForm Trial Management package—Shared by both Trial\_A and Trial\_B.
- **Trial\_A\_Clinical package**—Used only for Trial\_A.
- **Trial\_B\_Clinical package**—Used only for Trial\_B.



The contents of each trial-specific folder are set up to include the default reporting folder structure. The following illustration shows the contents of the Trial\_A folder.



This structure ensures that you can save trial-specific properties, such as prompt values and report schedules, for a specified report.

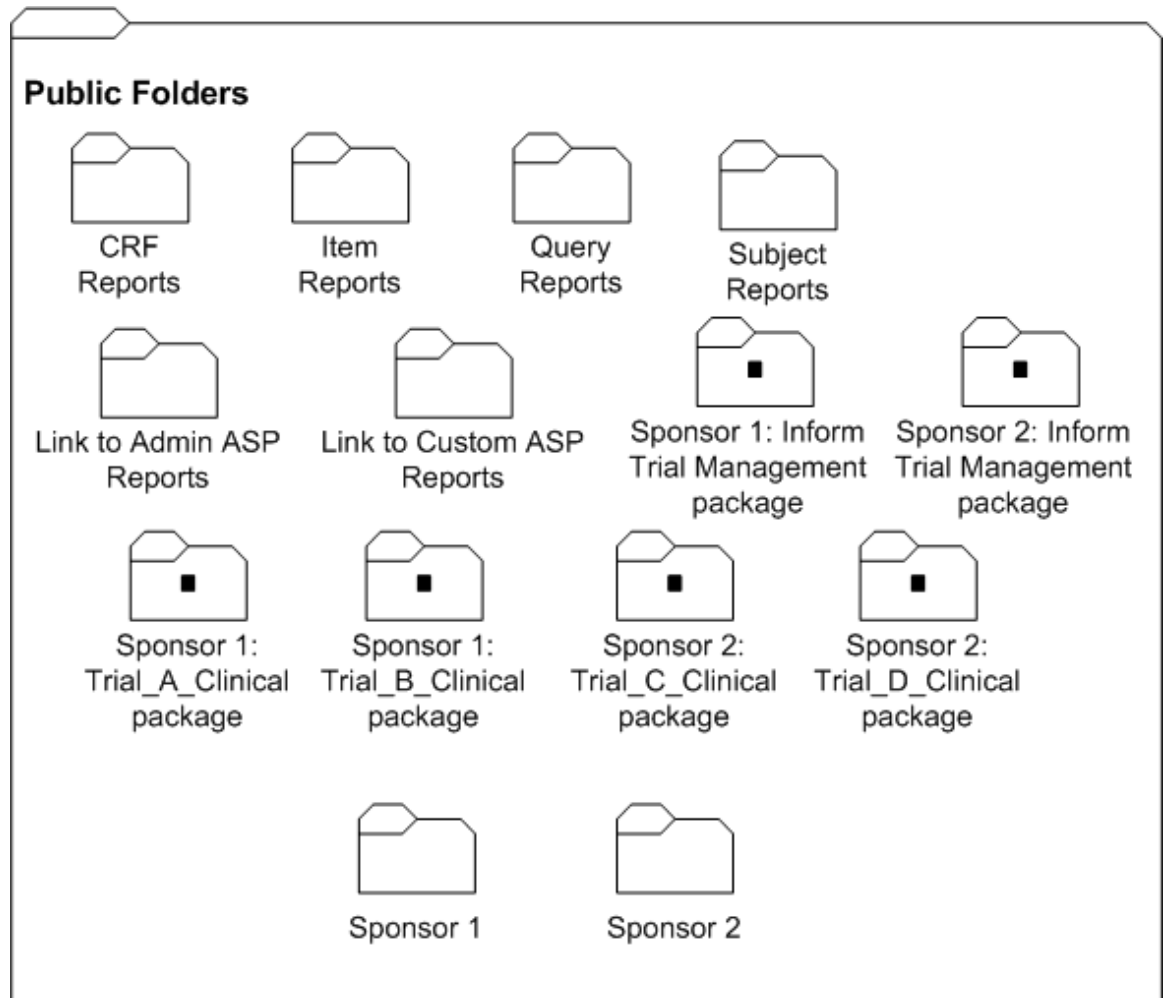
## Folder structure for multiple sponsors, multiple trials

To set up folders for several sponsors on one reporting server, consider creating subfolders for each sponsor under the **Public Folders** tab. You can only do this if you use InForm Report Folder Maintenance utility to perform all the configuration steps.

The following illustration shows how the **Public Folders** might look in the Reporting and Analysis portal with folders for two different sponsors, each hosting two different trials.

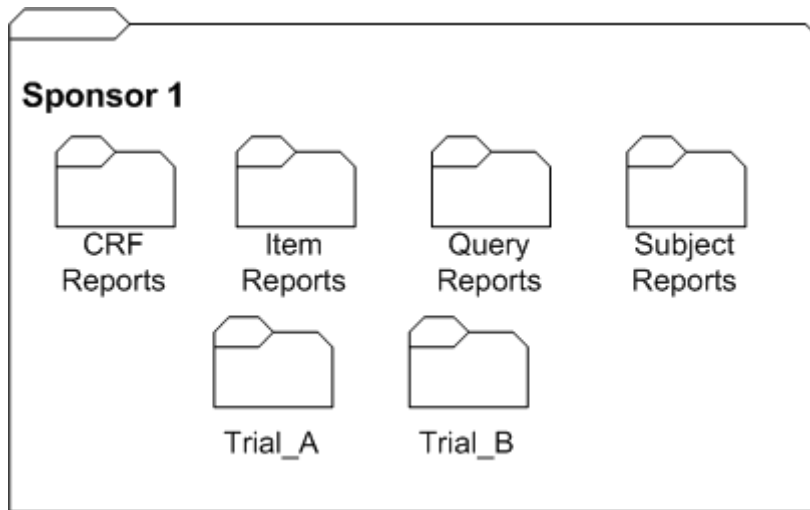
The following example shows:

- Four clinical packages: two for Sponsor 1 and two for Sponsor 2.
- Two InForm Trial Management packages: one for Sponsor 1 and one for Sponsor 2.

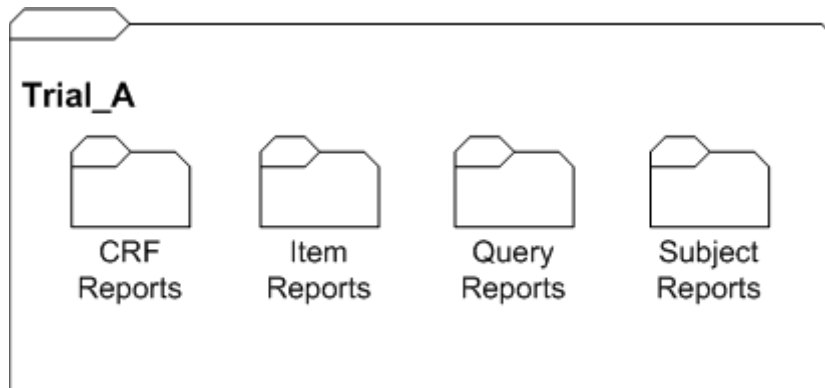




The contents of each sponsor-specific folder are set up to include the default reporting structure for a single trial. The following illustration shows the contents of the Sponsor 1 folder:



The Trial\_A folder contains all of the default standard report folders.



# Setting up a folder structure for multiple trials or sponsors

## Setting up reporting packages

All reporting packages must reside at the **Public Folders** level only; you cannot add a package to a subfolder. Therefore, if you are using several packages for different sponsors and trials, folders for each of these packages will appear in the **Public Folders** tab of the Reporting and Analysis portal.

### Trial-specific clinical package

Each trial uses a unique trial-specific clinical package. For details on how to publish a trial-specific clinical package, see the *Installation and Configuration Guide*.

### InForm Trial Management package

Trials that share a sponsor can share an InForm Trial Management package. This way, any revisions or updates to the package automatically apply to all trials for the sponsor.

You import the InForm Trial Management package by importing the Op model and Reports deployment archive. When you imported this archive as part of your installation, you imported both the standard reports folders and the InForm Trial Management package.

However, to be able to apply revisions to packages for individual trials, import a trial management package for each trial.

To import InForm Trial Management packages for different sponsors or trials:

- 1 Log on to the trial as the InForm system administrator.
- 2 Click **Reports**.
- 3 Select **Tools > Deployment**.
- 4 Click the **Import** tab.
- 5 In the **Op package and reports** entry, click **Set properties**.
- 6 Click the **Deployment** tab.
- 7 Click the **InForm Trial Management** checkbox. Make sure that all other options are deselected.
- 8 Click the pencil icon next to **InForm Trial Management**.
- 9 Change the name of the InForm Trial Management package to reflect the sponsor or trial that will be using the package.
- 10 Click **OK**.
- 11 Click **Import Now**.
- 12 Click **Finish**.

The Import screen appears.

## Creating new folders for multiple trials or sponsors

### Report package association

Each report is associated with the package (either the InForm Trial Management package or a trial-specific package) that was used to create it.

When you copy reports using the InForm Report Folder Maintenance utility, you can specify the package name to be used, if necessary. A report that you have copied to a new location must be associated with the package that will be used by the trial or sponsor who works with the report.

For reports that you created with the InForm Trial Management package:

- **Multiple trials, single sponsor**—You are not required to specify a new package name; the InForm Trial Management package is installed with every trial.
- **Multiple trials, multiple sponsors**—Each sponsor should provide an instance of the InForm Trial Management package; therefore, you may need to specify a new package name for the copied reports in the target location.

For reports that you created with the trial-specific clinical package, you must always specify a new package name for the copied reports in the target location.

### Report validation

The InForm Report Folder Maintenance utility validates copied reports against the associated packages when you copy the reports. Therefore, the packages must exist before you begin the copy operation. For more information, see *Setting up reporting packages* (on page 418).

The InForm Report Folder Maintenance utility stops whenever it encounters any error. You must correct the error, delete all objects within the target folder, and run the tool again to copy all reports. For example, although you can copy report definitions that contain clinical report elements from one trial to another, the schema of the clinical portion of the reporting database is unique to each trial; therefore, the clinical package that is generated for each trial is unique. If a clinical report contains a report element that does not exist in the target package, the report cannot be validated after it is copied.

### Report validation

The InForm Report Folder Maintenance utility validates copied reports against the associated packages when you copy the reports. Therefore, the packages must exist before you begin the copy operation. For more information, see *Setting up reporting packages* (on page 418).

The InForm Report Folder Maintenance utility stops whenever it encounters any error. You must correct the error, delete all objects within the target folder, and run the utility again to copy all reports. For example, although you can copy report definitions that contain clinical report elements from one trial to another, the schema of the clinical portion of the reporting database is unique to each trial; therefore, the clinical package that is generated for each trial is unique. If a clinical report contains a report element that does not exist in the target package, the report cannot be validated after it is copied.

## Report copy considerations summary

Source Package	Type of data in report	New package name required?	Report modifications required at the target location?
Trial-specific clinical package	Trial management data only.	Yes.*	No.
Trial-specific clinical package	Clinical data, or a mixture of clinical and trial management data.	Yes.*	Depends on the clinical packages and their elements. You may have to alter the report to ensure that it can be validated and copied to the target location.
<b>Single sponsor, multiple trials</b>			
InForm Trial Management package	Trial management data only.	No.	No.
<b>Multiple sponsors, multiple trials</b>			
InForm Trial Management package	Trial management data only.	Yes. Each sponsor folder should have its own InForm Trial Management package.	No.

\*The package must exist before you can copy the folders.

## Copying report folders

To copy report folders:

**InForm Report Folder Maintenance**

**Login Information**

User Name: supruser

Password: \*\*\*\*\*

LDAP Namespace: SuperUsers

**Source**

Copy from:  Standard report folders  Custom folders

Path: /content/folder[@name='report\_to\_exprt']

Folder prefix:

**Target**

Path: /content/folder[@name='cl316257067']

New folder prefix:

New folder name:

Package name: InForm Trial Management for cl316257067

Create Close Help

**Status**

Validating: Test - CRF Aging by Site Validation Completed! Updating the Reports... Report i

\*\*\*\* Copying completed successfully\*\*\*\*

User logoff in progress... Successful

++++++ Process Ended ++++++

- 1 On the reporting server, double-click the following file:  
`\crn\bin\PFMTRSetupUtil.exe`  
 The InForm Report Folder Maintenance window appears.
- 2 In the **Login Information** section, type the following information:
  - User name and password to log on to the reporting server.
  - Name of the LDAP namespace that is used for authentication by Sun One Directory Server.

- 3 In the **Source** section, enter the following information:
  - **Copy from**—Select the **Standard report folders** radio button to copy standard report folders, or select the **Custom folders** radio button to copy custom report folders.
  - **Path**—Type the path for the existing folder structure to copy. Do not include the folder name here. You can copy the source path from the source folder properties.
  - **Folder prefix**—Specify the prefix that identifies the folders to copy. For example, if you used T1 as a prefix for all folders belonging to Trial 1, specify T1 in the **Folder prefix** field to instruct the InForm Report Folder Maintenance utility to copy the folders with the T1 prefix.
- 4 In the **Target** section, enter the following information:
  - **Path**—Type the target path for the folder. Do not include the folder name.
  - **New Folder prefix**—Specify a prefix for the target folder name. For example, if you are creating the folder structure for Trial 2, you might type T2 as the prefix for the new folder.
  - **New Folder name**—Type the name of the target folder if it is different from the name of the source folder. You can specify a new folder name (the folder will be created) or an existing folder.
  - **Package name**—Type the name of the new package that should be associated with any trial-specific reports. Since each trial clinical package is unique, reports that contain clinical (trial-specific) data must be associated with a new clinical package when you copy them to a new location.

**Note:** The package must exist before you perform the copy.

- 5 Click **Create**.

The InForm Report Folder Maintenance utility begins to copy the folder structure to the new location. The **Status** section displays ongoing status and notifies you when the copy is complete.
- 6 On the InForm server, identify the top level reporting folder for the trial in the InForm Admin page.

## APPENDIX A

# Sample Data Import XML

### In this appendix

Overview .....	424
Importing screening and enrollment data.....	425
Importing new patient clinical data.....	426
Updating existing patient clinical data.....	427
Importing new itemset data.....	428
Editing an existing itemset.....	429
Deleting data from an itemset.....	430
Undeleting data from an itemset .....	431
Adding data to an unscheduled visit.....	432
Transferring patient records.....	433

## Overview

The import file for the MedML file option is an XML file that contains tags that specify the type of processing to perform during import, and the import data destinations and values. Cut and paste these samples and use any text editor that creates plain text files to edit the data to create your own import files.

- *Importing screening and enrollment data* (on page 425).
- *Importing new patient clinical data* (on page 426).
- *Updating existing patient clinical data* (on page 427).
- *Importing new itemset data* (on page 428).
- *Editing an existing itemset* (on page 429).
- *Deleting data from an itemset* (on page 430).
- *Undeleting data from an itemset* (on page 431).
- *Adding data to an unscheduled visit* (on page 432).
- *Transferring patient records* (on page 433).



## Importing screening and enrollment data

This sample file below contains the necessary data tags to import screening and enrollment data for patient XYZ at site PF.

```
<?xml version="1.0"?>
<CLINICALDATA>
<!--

Feature:Screen and Enroll
Description:This file sets up the user to test other XML features
-->

<!-- Screen Patient -->
<SCREEN SITEMNEMONIC="PF">
<DATA TAG="screen.0.patientinitials.patientinitials" VALUE="XYZ"/>
<DATA TAG="screen.0.eligible.eligible" VALUE="yes"/>
<DATA TAG="screen.0.datescreened.date" MONTH="1" DAY="6" YEAR="1999"/>
<DATA TAG="screen.0.dob.dob" MONTH="11" DAY="11" YEAR="1959"/>
</SCREEN>

<!-- Enroll Patient -->
<ENROLL PATIENTINITIALS="XYZ" SITEMNEMONIC="PF" PATIENTNUMBER="BK-XYZ"
ENROLL="TRUE">
<DATA TAG="consent.0.consentdate.date" MONTH="1" DAY="6" YEAR="1999"/>
<DATA TAG="consent.0.patientnumber.patientnumber" VALUE="BK-XYZ"/>
<DATA TAG="inclusion.0.age_inc.yesno" VALUE="1"/>
<DATA TAG="inclusion.0.hyper_inc.yesno" VALUE="1"/>
<DATA TAG="inclusion.0.understand_inc.yesno" VALUE="1"/>
<DATA TAG="inclusion.0.agree_inc.yesno" VALUE="1"/>
<DATA TAG="exclusion.0.secondary_ex.yesno" VALUE="0"/>
<DATA TAG="exclusion.0.malignant_ex.yesno" VALUE="0"/>
<DATA TAG="exclusion.0.allergyhistory_ex.yesno" VALUE="0"/>
<DATA TAG="exclusion.0.myocardial_ex.yesno" VALUE="0"/>
<DATA TAG="exclusion.0.monitor_ex.yesno" VALUE="0"/>
</ENROLL>
</CLINICALDATA>
```

## Importing new patient clinical data

The sample file below contains the necessary data tags to import clinical data to patient XYZ at site PF.

```
<?xml version="1.0"?>
<CLINICALDATA>

<!--
Feature:Form and Item comments
Description:This example demonstrates basic form and item comments
Requirements:PF_XYZ-Enroll.xml
-->

<!-- Demographics form -->
<PATIENTDATA
PATIENTINITIALS="XYZ"
SITEMNEMONIC="PF"
FORMSETREFNAME="Visit1"
FORMREFNAME="DEM"
COMMENT="This is the Demographics Form Comment">
<DATA TAG="DEM.0.GENDER.GENDERRADIO" VALUE="1" COMMENT="Gender Comment"/>
<DATA TAG="DEM.0.DEMDOB.dob" MONTH="2" DAY="14" YEAR="1961" COMMENT="Date Of
Birth Comment"/>
<DATA TAG="DEM.0.RACE.RACEGROUP" CHILDSELECTED="RACETEXT" COMMENT="Race Text
Comment"/>
<DATA TAG="DEM.0.RACE.RACEGROUP.RACETEXT" VALUE="10k" COMMENT="Race Text Value
Comment"/>
<DATA TAG="DEM.0.HEIGHT.PFHT_TC" VALUE="74" UNIT="Inches" COMMENT="Height
Comment"/>
<DATA TAG="DEM.0.WRISTCIRC.PFWC_TC" VALUE="6.0" UNIT="Inches" COMMENT="Height
Unit Comment"/>
<DATA TAG="DEM.0.FRAME.FRAME_CC" VALUE="1"/>
<DATA TAG="SH.0.SMOKE.SMOKERADIO" VALUE="Y"/>
<DATA TAG="SH.0.EVERSMOKED.SMOKERADIO" VALUE="N"/>
<DATA TAG="SH.0.WHATSMOKED.SMOKEGROUPRADIO" CHILDSELECTED="SMOKEGROUP"/>
<DATA TAG="SH.0.WHATSMOKED.SMOKEGROUPRADIO.SMOKEGROUP" VALUE="SMOKES"/>
<DATA TAG="SH.0.WHATSMOKED.SMOKECHECKBOX" VALUE="cigarette,pipe"/>
<DATA TAG="SH.0.HOWMUCHSMOKED.SMOKERADIO2" CHILDSELECTED="NUMTEXT"/>
<DATA TAG="SH.0.HOWMUCHSMOKED.SMOKERADIO2.NUMTEXT" VALUE="10"/>
<DATA TAG="SH.0.YRSSMOKED.SMOKERADIO2" VALUE="NDElement"/>
</PATIENTDATA>

</CLINICALDATA>
```

## Updating existing patient clinical data

The sample file below contains the necessary data tags to edit or clear existing data about a patient.

```
<?xml version="1.0"?>
<CLINICALDATA>

<!--
Feature:Edit an existing patient's form record
Description:This example demonstrates editing one item, then clearing an item's
value
Requirements:The following must be run prior to this script
PF_XYZ-Enroll.xml
PF_XYZ-DEMANDcommentts.xml
-->

<!-- Demographics form -->
<EDITPATIENTDATA
PATIENTINITIALS="XYZ"
SITEMNEMONIC="PF"
FORMSETREFNAME="Visit1"
FORMREFNAME="DEM"
REASONOTHER="updated data">
<DATA TAG="DEM.0.HEIGHT.PFHT_TC" VALUE="75" UNIT="Inches" />
</EDITPATIENTDATA>

<!-- Demographics form -->
<EDITPATIENTDATA
PATIENTINITIALS="XYZ"
SITEMNEMONIC="PF"
FORMSETREFNAME="Visit1"
FORMREFNAME="DEM"
REASONOTHER="updated data">
<DATA TAG="DEM.0.WRISTCIRC.PFWC_TC" CLEARVALUE="TRUE" />
</EDITPATIENTDATA>
</CLINICALDATA>
```

## Importing new itemset data

The sample file below contains the necessary data tags to import new itemset data.

```
<?xml version="1.0"?>
<CLINICALDATA>

<!--
Feature: Submitting Itemset Data
Description: This example demonstrates to submitting itemset data
Requirements: The following must be run prior to this script
PF_XYZ-Enroll.xml
-->

<!-- non explicit itemset index (append to existing itemsets if exists) -->
<PATIENTDATA
PATIENTINITIALS="ITM"
SITEMNEMONIC="PF"
FORMSETREFNAME="Visit1"
FORMREFNAME="HH"
SECTIONNAME="PT"
ITEMSETNAME="PT"
<DATA TAG="PT.PT.THERAPYTEXT.THERAPYTEXT" VALUE="aspirin"/>
<DATA TAG="PT.PT.DOSAGETEXT.DOSAGETEXT" VALUE="1 tablet"/>
<DATA TAG="PT.PT.DOSEDATE.DOSEDATE" MONTH="12" DAY="23" YEAR="1998"/>
<DATA TAG="PT.PT.DISCULLDOWN.DISCULLDOWN" VALUE="Not Effective"/>
</PATIENTDATA>

<!-- Create at itemset index 2 -->
<PATIENTDATA
PATIENTINITIALS="ITM"
SITEMNEMONIC="PF"
FORMSETREFNAME="Visit1"
FORMREFNAME="HH"
SECTIONNAME="PT"
ITEMSETNAME="PT"
COMMENT="ItemSet #2 - explicit index of 2">
<DATA TAG="PT.PT.THERAPYTEXT.THERAPYTEXT" VALUE="aspirin3"/>
<DATA TAG="PT.PT.DOSAGETEXT.DOSAGETEXT" VALUE="1 tablet"/>
<DATA TAG="PT.PT.DOSEDATE.DOSEDATE" MONTH="12" DAY="23" YEAR="1998"/>
<DATA TAG="PT.PT.DISCULLDOWN.DISCULLDOWN" VALUE="Not Effective"/>
</PATIENTDATA>

<!-- Force itemset index 3 -->
<PATIENTDATA
PATIENTINITIALS="ITM"
SITEMNEMONIC="PF"
FORMSETREFNAME="Visit1"
FORMREFNAME="HH"
SECTIONNAME="PT"
ITEMSETNAME="PT"
ITEMSETINDEX="3">
<DATA TAG="PT.PT.THERAPYTEXT.THERAPYTEXT" VALUE="Bucket"/>
<DATA TAG="PT.PT.DOSAGETEXT.DOSAGETEXT" VALUE="1 tablet"/>
<DATA TAG="PT.PT.DOSEDATE.DOSEDATE" MONTH="12" DAY="23" YEAR="1998"/>
<DATA TAG="PT.PT.DISCULLDOWN.DISCULLDOWN" VALUE="Not Effective"/>
</PATIENTDATA>

<!-- Create at itemset index 4 -->
<PATIENTDATA
PATIENTINITIALS="ITM"
SITEMNEMONIC="PF"
FORMSETREFNAME="Visit1"
FORMREFNAME="HH"
SECTIONNAME="PT"
ITEMSETNAME="PT"
<DATA TAG="PT.PT.THERAPYTEXT.THERAPYTEXT" VALUE="To Delete"/>
<DATA TAG="PT.PT.DOSAGETEXT.DOSAGETEXT" VALUE="To Delete"/>
<DATA TAG="PT.PT.DOSEDATE.DOSEDATE" MONTH="12" DAY="23" YEAR="1998"/>
<DATA TAG="PT.PT.DISCULLDOWN.DISCULLDOWN" VALUE="Not Effective"/>
</PATIENTDATA>

</CLINICALDATA>
```

## Editing an existing itemset

The sample file below contains the necessary data tags to edit a row in an existing itemset.

```
<?xml version="1.0"?>
<CLINICALDATA>

<!--
Feature:Edit and Itemset item
Description:This example demonstrates editing an itemset item
Requirements:The following must be run prior to this script
PF_XYZ-Enroll.xml
PF_XYZ-ItemSet.xml
-->

<!-- explicit itemset index -->
<EDITPATIENTDATA
PATIENTINITIALS="XYZ"
SITEMNEMONIC="PF"
FORMSETREFNAME="Visit1"
FORMREFNAME="HH"
SECTIONNAME="PT"
ITEMSETNAME="PT"
ITEMSETINDEX="2"
REASONOTHER="test reason"
<DATA TAG="PT.PT.THERAPYTEXT.THERAPYTEXT" VALUE="aspirinEdit"
COMMENT="ItemLevelComment"/>
</EDITPATIENTDATA>

</CLINICALDATA>
```

## Deleting data from an itemset

The sample file below contains the necessary data tags to delete data from an existing itemset.

```
<?xml version="1.0"?>
<CLINICALDATA>

<!--
Feature:Delete an Itemset record (DELETEITEMSET tag)
Description:This example demonstrates to Delete an existing itemset record
Requirements:The following must be run prior to this script
PF_XYZ-Enroll.xml
PF_XYZ-ItemSet.xml
PF_XYZ-ItemSetDelete.xml
-->

<!-- non explicit itemset index (append to existing itemsets if exists) -->
<EDITPATIENTDATA
PATIENTINITIALS="XYZ"
ITEMNEMONIC="PF"
FORMSETREFNAME="Visit1"
FORMREFNAME="HH"
SECTIONNAME="PT"
ITEMSETNAME="PT"
DELETEITEMSET="TRUE"
ITEMSETINDEX="4"
REASONOTHER="test reason"
</EDITPATIENTDATA>

</CLINICALDATA>
```

## Undeleting data from an itemset

The sample file below contains the necessary data tags to recover deleted data from an itemset.

```
<?xml version="1.0"?>
<CLINICALDATA>

<!--
Feature:Undelete an Itemset record (UNDELETEITEMSET tag)
Description:This example demonstrates to Undelete a previously deleted itemset.
Requirements:The following must be run prior to this script
PF_XYZ-Enroll.xml
PF_XYZ-ItemSet.xml
PF_XYZ-ItemSetDelete.xml
-->

<EDITPATIENTDATA
PATIENTINITIALS="XYZ"
SITEMNEMONIC="PF"
FORMSETREFNAME="Visit1"
FORMREFNAME="HH"
SECTIONNAME="PT"
ITEMSETNAME="PT"
UNDELETEITEMSET="TRUE"
ITEMSETINDEX="4"
REASONOTHER="test reason"
</EDITPATIENTDATA>

</CLINICALDATA>
```

## Adding data to an unscheduled visit

The sample file below contains the necessary data tags to add data to an unscheduled visit for a particular patient.

```
<?xml version="1.0"?>
<CLINICALDATA>

<!--
Feature:Unscheduled Visits
Description:This example demonstrates the sequence of XML submits to use
Unscheduled Visits
Requirements:PF_XYZ-Enroll.xml
-->

<!-- DOV form -->
<PATIENTDATA
PATIENTINITIALS="XYZ" SITEMNEMONIC="PF"
FORMSETREFNAME="UnschVisit" FORMREFNAME="DOV" NEWUNSCHEDVISIT="TRUE">
<DATA TAG="DOV.0.DOV.DOV" MONTH="2" DAY="1" YEAR="1999"/>
</PATIENTDATA>

<!-- VitalSigns form -->
<PATIENTDATA
PATIENTINITIALS="XYZ" SITEMNEMONIC="PF" FORMSETINDEX="1"
FORMSETREFNAME="UnschVisit" FORMREFNAME="VS">
<DATA TAG="VS.0.DATEASSESS.COMMONDATE" MONTH="3" DAY="1" YEAR="1999"/>
<DATA TAG="VS.0.WEIGHT.PFWT_TC" VALUE="150" UNIT="Pound"/>
<DATA TAG="VS.0.TEMPTEXT.TEMPTEXT" VALUE="98.7" UNIT="Fahrenheit"/>
<DATA TAG="VS.0.BPREADING.BPREADINGGROUP.SYSTEXT" VALUE="130"/>
<DATA TAG="VS.0.BPREADING.BPREADINGGROUP.DIASTEXT" VALUE="85"/>
</PATIENTDATA>

<!-- DOV form -->
<PATIENTDATA
PATIENTINITIALS="XYZ" SITEMNEMONIC="PF"
FORMSETREFNAME="UnschVisit" FORMREFNAME="DOV" NEWUNSCHEDVISIT="TRUE">
<DATA TAG="DOV.0.DOV.DOV" MONTH="2" DAY="2" YEAR="1999"/>
</PATIENTDATA>

<!-- VitalSigns form -->
<PATIENTDATA
PATIENTINITIALS="XYZ" SITEMNEMONIC="PF" FORMSETINDEX="2"
FORMSETREFNAME="UnschVisit" FORMREFNAME="VS">
<DATA TAG="VS.0.DATEASSESS.COMMONDATE" MONTH="3" DAY="2" YEAR="1999"/>
<DATA TAG="VS.0.WEIGHT.PFWT_TC" VALUE="150" UNIT="Pound"/>
<DATA TAG="VS.0.TEMPTEXT.TEMPTEXT" VALUE="98.7" UNIT="Fahrenheit"/>
<DATA TAG="VS.0.BPREADING.BPREADINGGROUP.SYSTEXT" VALUE="130"/>
<DATA TAG="VS.0.BPREADING.BPREADINGGROUP.DIASTEXT" VALUE="85"/>
</PATIENTDATA>

</CLINICALDATA>
```



## Transferring patient records

This example shows the tags in a MedML file that is used to transfer several patients. Note that each pair of PATIENTSITECHANGE tags defines current and destination site information for one patient.

```
<?xml version="1.0"?>
<CLINICALDATA>
 <PATIENTSITECHANGE REASON="new patient address">
 <NEWSITE SITEMNEMONIC="MCLEAN" />
 <CURRENTSITE SITEMNEMONIC="PF" PATIENTNUMBER="1003"/>
 </PATIENTSITECHANGE>
<!--For patient 1001 the file specifies a new patient number because a patient already exists at the
McLean Hospital site with the same patient number.-->

 <PATIENTSITECHANGE REASON="new patient address">
 <NEWSITE SITENAME="McLean Hospital" PATIENTNUMBER="1002"/>
 <CURRENTSITE SITEMNEMONIC="PF" PATIENTNUMBER="1001"/>
 </PATIENTSITECHANGE>
<!--The DUPLICATEORDER tag indicates that patient DDD is the second patient with those
initials to be screened at the Oracle site.-->

 <PATIENTSITECHANGE REASON="new patient address">
 <NEWSITE SITEMNEMONIC="MCLEAN" />
 <CURRENTSITE SITENAME="Oracle" PATIENTINITIALS="DDD"
 DUPLICATEORDER="2" />
 </PATIENTSITECHANGE>
</CLINICALDATA>
```