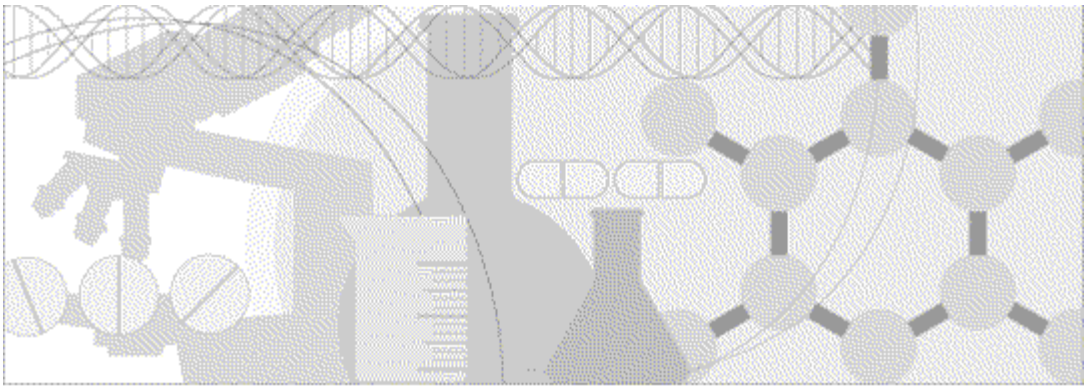
# Utilities Guide

## Oracle® Health Sciences InForm 6.0



ORACLE

Part number: E37413-01

# Contents

# About this guide

## In this preface

# Overview of this guide

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

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

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.

## Audience

This guide is for study designers and system administrators who need to move data into and out of the InForm database, modify existing InForm data, and monitor InForm processes and utilities.

# Related information

## Documentation

All documentation is available from the Oracle Software Delivery Cloud (https://edelivery.oracle.com) and the Download Center (https://extranet.phaseforward.com).

| Document | Description |
|---|---|
| *Release Notes* | The *Release Notes* document describes enhancements introduced and problems fixed in the current release, upgrade considerations, release history, and other late-breaking information. |
| *Known Issues* | The *Known Issues* document provides detailed information about the known issues in this release, along with workarounds, if available.<br><br>**Note:** The most current list of known issues is available on the Extranet. To sign in to the Extranet, go to https://extranet.phaseforward.com. |
| *Upgrade and Migration Guide* | The *Upgrade and Migration Guide* provides instructions for upgrading and migrating the InForm software and InForm Portal software to the current InForm release, and for upgrading the Cognos 8 Business Intelligence software for use with the Reporting and Analysis module. The guide also describes any changes and additions made to the database schema, MedML, and resource files. |
| *Secure Configuration Guide* | The *Secure Configuration Guide* 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. |
| *Installation Guide* | The *Installation Guide* describes how to install the software and configure the environment for the InForm application and Cognos 8 Business Intelligence software. |
| *Study and Reporting Setup Guide* | The *Study and Reporting Setup Guide* describes how to perform the tasks that are required to set up an InForm study and configure the Reporting and Analysis module for the study. |
| *User Guide* | The *User Guide* provides an overview of the InForm application including details on multilingual studies, how to navigate through the user interface, and how to use the application to accomplish typical tasks you perform while running a clinical study.<br><br>This document is also available from the Documentation CD and the InForm user interface. |

| Document | Description |
|---|---|
| *Reporting and Analysis Guide* | The *Reporting and Analysis Guide* 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 study 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.<br><br>This document is also available from the Documentation CD. |
| *Reporting Database Schema Guide* | The *Reporting Database Schema Guide* describes the Reporting and Analysis database schema, and provides information on creating Reporting Database Extracts (RDEs). |
| *Portal Administration Guide* | The *Portal Administration Guide* provides step-by-step instructions for setting up the InForm Portal software, and configuring and managing the InForm Portal application.<br><br>This document is also available from the Documentation CD. |
| *Utilities Guide* | The *Utilities Guide* provides information about and step-by-step instructions for using the following utilities:<br><br>• PFConsole utility<br>• MedML Installer utility<br>• InForm Data Import utility<br>• InForm Data Export utility<br>• InForm Performance Monitor utility<br>• InForm Report Folder Maintenance utility<br><br>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.<br><br>This document is also available from the Documentation CD. |
| MedML Installer utility online Help | 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 study components into the InForm database.<br><br>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.<br><br>This document is also available from the user interface. |

| Document | Description |
|---|---|
| InForm Data Export utility online Help | 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:<br>• Customer-defined database (CDD).<br>• Name value pairs.<br>This document is also available from the user interface. |
| InForm Data Import utility online Help | 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.<br>This document is also available from the user interface. |

# 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.

CHAPTER 1

# PFConsole utility

## In this chapter

# Overview of the PFConsole utility

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

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

**Note:** Oracle recommends that you use the PFConsole utility to run these utilities from the command line, and that you do not run them individually in the command line window.

# Running the PFConsole utility from the command line

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*—MedML Installer utility.
  - *pfimport*—InForm Data Import utility.
  - *pfexport*—InForm Data Export utility.
- *-autorun* is a required parameter of the PFConsole utility.
- <*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 353).
- ***Running the InForm Data Export utility from the command line*** (on page 365).

**Note:** All command line applications, such as the MedML Installer utility, the InForm Data Import utility, and the InForm Data Export utility use the default product locale specified during the InForm installation, or through PFAdmin commands.

## Example

```
pfconsole pfmminst -trial pf206 -verbose -autorun -outfile text.log -xml
filename.xml
```

# Using PFConsole utility within a script

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

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

where:

<*pfconsole command line*> is the string of commands you specify when you run the PFConsole utility.

For more information, see ***Running the PFConsole utility from the command line*** (on page 3).

CHAPTER 2

# MedML and the MedML Installer utility

## In this chapter

# MedML and the MedML schema

## Creating an XML file

You can use any text editor, such as Notepad or TextPad, to create and modify XML files.

**Note:** Oracle suggests that you save the sample XML files that are delivered as part of the InForm installation media under a new name and edit those files. The files are available at: \\<*Installation_Directory*>\InForm \Sample_PFST50\Admin.

To create an XML file for MedML component definitions for administrative data:

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 component and include the appropriate version string as the value of the xmlns attribute:

```
<MEDMLDATA xmlns="PhaseForward-MedML-Inform4"> </MEDMLDATA>
```

3 Between the MedMLDATA start and end elements, enter definitions for the following types of resources:

- Language
- Browser
- GIF, JPG, or text resource files
- HTML template
- Report
- Documentation
- StudyVersionDoc

4 Install study data definitions in the following order:

a SysConfig

b SequenceType

c Sequence

d Right

e GroupType

f User

g UserImage

h Site, including references to users who serve as site contacts, if applicable.

i Sponsor, including references to users who serve as sponsor contacts, if applicable.

j QueryGroup, including references to users who are members of the group.

k SignatureGroup, including references to users who are members of the group.

l SiteGroup, including references to users who have access to the site.

m RightsGroup, including references to the rights that make up the group and references to users who are assigned the rights.

n SignCRF

5 Build a StudyVersion that specifies the study protocol version.

6 Create StudyVersionSite definitions that specify which study version each site is using.

## Component definitions

An XML component definition is the basic building block of an XML file. Each component definition is made up of elements and attributes. Component definitions have a hierarchical relationship to each other (that is, a component can have child components).

## XML elements

An element consists of the component name surrounded by angle brackets (< >). For example:

```
<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>
```

When you define a compound component (a component with children), the parent component has both opening and closing elements, and the child components have an opening element that ends with a slash. For example:

```
<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>
```

Components with no children have no separate closing element. Instead, the component definition ends with a slash and a closing angle bracket, following any attributes that you specify. For example:

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

## XML attributes

A set of attributes uniquely identifies the component and describes how the component appears online. 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.

The following example shows 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.

## Child components

Child components, or children, in a component definition allow 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, Controlref) 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 element of the parent component, after the parent component attributes, as shown in the following example.

```
<QUERYGROUP GROUPNAME="CRA Query">
    <USERREF USERNAME="Kevin"/>
    <USERREF USERNAME="George"/>
</QUERYGROUP>
```

### Ordering component definitions

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

The order in which component definitions are created is important when you load the component definitions into the database 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.

You can organize study component definitions in any way that is convenient, as long as you follow the component order described in *Creating an XML file* (on page 6), and you can create multiple study component definition XML files. If you separate study component definitions into multiple files, load them in a sequence that maintains the prescribed component order.

## Working with RSP files

Response files, which are text files with the RSP extension, list XML file names in the order in which they should be processed. You can create or modify RSP files using any text editor.

You might want to use or modify RSP files when you load multiple files at the same time, for example, when you:

- Load a subset of components from a study.
- Load base components from a library of standard study components.
- Prepare to deploy a study (the RSP file should reference all relevant files).
- Perform mid-study changes (the RSP file should include only those study components that have changed).

You can use an RSP file to load all the files for a study by specifying the paths for the XML files.

To create RSP files, from a Command Prompt window, in the directory containing the XML files, type:

```
dir *.xml /b > RSPfileName.rsp
```

This command creates an RSP file in the folder in which you executed the command. The resulting RSP file contains all the XML files in the current directory in alphabetical order.

**Note:** You might have to rearrange the order of the files in the RSP file if there are dependencies. To exclude an XML file from the RSP file, remove the file or comment it out in the RSP file using the pound sign (#).

## RefNames

RefNames are names that are used to identify component definitions. In a Form definition window, the hierarchical representation of a form and its components shows each component by RefName. RefNames are also used to uniquely identify the path to a specific control when attaching a rule definition.

The following rules apply to the use of RefNames:

- RefNames can have a maximum of 63 characters.
- RefNames must be unique within a study and component type. For example, no two item definitions in a study could have the same RefName. However, a CRF and section can have the same RefName.
- RefNames are case-sensitive.
- After a definition containing a RefName is installed in the study database, you cannot change the RefName. After this point, to create a new RefName, you must create a new object.
- The following terms are disallowed for use in RefNames because they will cause a failure during the Reporting and Analysis database set up:
    - CD_COUNT
    - AFROWID
    - SUBJECTID
    - SITEID
    - STUDYVERSIONID
    - SUBJECTVISITID
    - SUBJECTVISITREV
    - VISITID
    - VISITINDEX
    - FORMID
    - FORMREV
    - FORMINDEX
    - SUBJECTINITIALS
    - SITEMNEMONIC
    - VISITMNEMONIC
    - FORMMNEMONIC
    - VISITORDER
    - SITENAME
    - SITECOUNTRY
    - SECTIONID

- ITEMSETID
- ITEMSETINDEX
- ITEMSETIDX
- DELETEDITEM
- DELETEDFORM
- FORMIDX

## Reserved words for Reporting and Analysis

The following reserved words cannot be used for the Short Question for an item in the Central Designer application. The Short Question text becomes the itemset column header in the Reporting and Analysis module. If you use the following words, the Reporting and Analysis clinical model cannot be created.

- Subject Initials
- Subject Number
- Visit Mnemonic
- Form Mnemonic
- Site Mnemonic
- Visit Index
- Visit Order
- DOV
- Deleted Form
- CREATEDBYUSERID
- CREATEDDATETIME
- MODIFIEDBYUSERID
- MODIFIEDDATETIME
- Site Name
- Site Country
- SUBJECTID
- SITEID
- SUBJECTVISITID
- VISITID
- VISITINDEX
- FORMID
- FORMREV
- FORMINDEX
- CD_COUNT

# UUIDs

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

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, to allow users to define subject 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 subject number data-entry field.

To change a UUID, you must install the change using the MedML Installer in nonstrict mode. For more information, see *About the MedML Installer window* (on page 36).

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

The following cases require a specific well-known UUID:

- Common forms
- Date Of Visit
- Enrollment forms
- Group types
- Subject ID form
- Subject Initials item
- Subject Number item
- Randomization
- RegDocs
- Repeating visits
- Screening forms
- Sequences
- Study Completion
- Visit Report

## Common CRFs

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

You define a common CRF by including it in a COMMONCRF formset in the StudyVersion definition.

**Note:** After you define a common CRF and add data for any subject to the CRF, you must not change the form type to a regular CRF during a study version change. Similarly, you must not change a regular CRF to a common CRF if the form includes subject data. Doing so will cause data loss.

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 cannot 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 definition.

To define a common CRF, use the following UUID in the definition of the COMMONCRF formset in which the CRF is included:

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

## Date Of Visit

The Date of Visit item 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 datetime control 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 |

**Note:** Do not use a common CRF 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

| Component | UUID |
|---|---|
| Enrollment formset | d882ce3a-0f42-11d2-a419- 00a0c963e0ac |
| Enrollment form | d882ce3b-0f42-11d2-a419- 00a0c963e0ac |
| Enrollment form section that contains the Subject Number data entry item | abcfa388-223a-11d2-a426- 00a0c963e0ac |

Additionally, if the Enrollment form includes an editable Subject Number item, that item requires a well-known UUID. For more information, see *Subject Number item* (on page 16).

## Group types

The GROUPTYPE element 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:

| 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 |

## Subject ID form

Use the Subject ID form to allow users to change the Subject ID after a subject is fully enrolled. The Subject ID form requires the following UUIDs:

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

## Subject Initials item

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

| Form component | UUID |
|---|---|
| Subject Initials item | aeb64f16-127c-11d2-a41c-00a0c963e0ac |
| Subject Initials text control | aeb64f17-127c-11d2-a41c-00a0c963e0ac |
| Date of Birth item | 96cae359-126c-11d2-a41c- 00a0c963e0ac |
| Date of Birth control | 40aee712-217c-11d2-a425- 00a0c963e0ac |
| Date Screened item | 96cae356-126c-11d2-a41c- 00a0c963e0ac |
| Date Screened control | 96cae357-126c-11d2-a41c- 00a0c963e0ac |

## Subject Number item

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

| Form component | UUID |
|---|---|
| Subject Number item | 3d25ee4c-7f1b-11d2-a728- 00a0c977c64b |
| Subject Number text control | 433daff6-7f1c-11d2-a728-00a0c977c64b |
| Enrollment form section that contains the Subject Number data-entry item | abcfa388-223a-11d2-a426- 00a0c963e0ac |

## Randomization

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

| Form component | UUID |
|---|---|
| RANDOMIZATION calculated control | 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 Regulatory Document CRF. The definition of Reg Docs requires the following UUIDs:

| Component | UUID |
|---|---|
| Regulatory Document formset | PF_UUID_REGDOCS_FORMSET |
| Regulatory Document form | PF_UUID_REGDOCS_FORM |
| Regulatory Document section | PF_UUID_REGDOCS_SECTION |

## Repeating visits

A FORMSET definition with the **REPEATING** attribute set to true allows you to define an unscheduled visit with its own set of CRFs. To set up a visit, you must use the following UUIDs to create a section, item, and datetime control 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 CRFs

The definition of a Screening CRF 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 Subject Initials, Date of Birth, Date Screened | 96cae354-126c-11d2-a41c- 00a0c963e0ac |

## Sequences

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

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

## Study Completion

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

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

| Component | UUID |
|---|---|
| Visit in which the Study Completion CRF is included | F4699051-69E2-11D2-8FB5-00A0C977C66A |
| Study Completion CRF | 7314A6A5-316E-11d2-8F9A-00A0C977C66A |
| Control to indicate completion status | PF_SC_COMPLETECTL |
| Element objects defining the values and display text for study completion or noncompletion | Both of the following element objects must be present in the study definition:<br>• PF_SC_STUDY_COMPLETE — Indicates that the subject completed the study.<br>• PF_SC_STUDY_INCOMPLETE — Indicates that the subject did not complete the study. |
| Control to indicate the reason the subject dropped out of the study | PF_SC_REASONCTL |
| Text resources mapping the internal values of noncompletion reasons to the text of column headings in the Reporting and Analysis module | PF_SC_REASONCTL_*internal_reason_value*<br>*Internal_reason_value* is one of the following:<br>• The Value property defined for an element object in a simple or pull-down control included in the control specifying noncompletion reason.<br>• The Selection Value property defined for any other type of subordinate control included in the control specifying noncompletion reason.<br>• The default value for a subordinate control for which no Selection Value property has been defined. The default value is assigned by the InForm application and is in the format !pf! *control_DBUID_path*. |

### 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

### <!– –> (Comment)

The comment element 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 within another element.

#### Example

```
<?xml version="1.0"?>
<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 study component definitions; for example, study protocols, CRF Help, CRF questions and notes, and rule help.

Because angle brackets (greater than and less than symbols) are disallowed by the MedML Installer utility, you must use the HTML escape character equivalents when installing data from a MedML file to the InForm application.

For more information, see *HTML special characters* (on page 22) and *Disallowed characters* (on page 24).

| For this formatting | Use these equivalent tags | As substitutes for these HTML tags |
|---|---|---|
| Bold text | <B><br></B> | <B><br></B> |
| Line break | <BR> | <BR> |
| Centering text an equal distance from the left and right edges of the document | <CENTER><br></CENTER> | <CENTER><br></CENTER> |
| Italic text | <I><br></I> | <I><br></I> |
| List items | <LI><br></LI> | <LI><br></LI> |
| Ordered (numbered) lists | <OL><br></OL> | <OL><br></OL> |
| Paragraphs | <P><br></P> | <P><br></P> |
| Preformatted plain text; for example, computer output | <PRE><br></PRE> | <PRE><br></PRE> |
| Strikethrough text | <S><br></S> | <S><br></S> |
| Strikethrough text | <STRIKE><br></STRIKE> | <STRIKE><br></STRIKE> |
| Subscript text | <SUB><br></SUB> | <SUB><br></SUB> |
| Superscript text | <SUP><br></SUP> | <SUP><br></SUP> |

| For this formatting | Use these equivalent tags | As substitutes for these HTML tags |
|---|---|---|
| Monospace font | <TT> | <TT> |
| | </TT> | </TT> |
| Underlined text | <U> | <U> |
| | </U> | </U> |
| Unordered (bulleted) lists | <UL> | <UL> |
| | </UL> | </UL> |

## HTML special characters

The HTML specification includes character sequences for specifying special characters. When you include HTML special characters in a study 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.

The following special character definitions are supported by the PDF output format, in which the InForm Data Export utility exports printable CRFs.

| Character | Entity Name | Numeric Code | Descriptive |
|---|---|---|---|
| " | quotation mark | &#34; | &quot; |
| & | ampersand | &#38; | &amp; |
| < | less-than sign | &#60; | &lt; |
| > | greater-than sign | &#62; | &gt; |
| | non-breaking space |   |   |
| ¡ | inverted exclamation | &#161; | &iexcl; |
| ¢ | cent sign | &#162; | &cent; |
| £ | pound sterling | &#163; | &pound; |
| ¤ | general currency sign | &#164; | &curren; |
| ¥ | yen sign | &#165; | &yen; |
| § | section sign | &#167; | &sect; |
| ¨ | umlaut(dieresis) | &#168; | &uml; |
| © | copyright | &#169; | &copy; |
| ª | feminineordinal | &#170; | &ordf; |
| « | left angle quote, guillemotleft | &#171; | &laquo; |
| ¬ | not sign | &#172; | &not; |
| - | soft hyphen | &#173; | &shy; |

| Character | Entity Name | Numeric Code | Descriptive |
|---|---|---|---|
| ® | 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 |
| Ü | 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 |

| Character | Entity Name | Numeric Code | Descriptive |
|---|---|---|---|
| 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<br>• Form title<br>• Question text<br>• Query answer text<br>• Query text |
| Single quote | • Question text<br>• Query answer text<br>• Query text<br>• Site name |
| Apostrophe (') | • Form title<br>• Question text |
| \ or \\ | Do not use these characters. |

| Character | Where not to use |
|---|---|
| > or < | Anywhere; use HTML escape character equivalents:<br>• > -- > or ><br>• < -- < or < |
| Superscript and subscript formatting specifications | Elements that will be used in pulldown controls |

Additionally, avoid copying and pasting from the Microsoft Word application into text boxes and query text, as the Microsoft Word application can change characters to Unicode format.

## Locale codes

The following table lists the locale codes that you can use to specify translated text in the definition of **TRANSLATION** attributes. Note that the codes used in the Central Designer application are culture-specific. In the table, culture-specific codes have a hyphen between the language and culture.

For more information, see *Translation* (on page 211).

| Culture and language code | Culture |
|---|---|
| af-ZA | Afrikaans (South Africa) |
| sq-AL | Albanian (Albania) |
| ar-DZ | Arabic (Algeria) |
| ar-BH | Arabic (Bahrain) |
| ar-EG | Arabic (Egypt) |
| ar-IQ | Arabic (Iraq) |
| ar-JO | Arabic (Jordan) |
| ar-KW | Arabic (Kuwait) |
| ar-LB | Arabic (Lebanon) |
| ar-LY | Arabic (Libya) |
| ar-MA | Arabic (Morocco) |
| ar-OM | Arabic (Oman) |
| ar-QA | Arabic (Qatar) |
| ar-SA | Arabic (Saudi Arabia) |
| ar-SY | Arabic (Syria) |
| ar-TN | Arabic (Tunisia) |
| ar-AE | Arabic (U.A.E.) |
| ar-YE | Arabic (Yemen) |
| hy-AM | Armenian (Armenia) |

| Culture and language code | Culture |
|---|---|
| az-Cyrl-AZ | Azeri (Azerbaijan, Cyrillic) |
| az-Latn-AZ | Azeri (Azerbaijan, Latin) |
| eu-ES | Basque (Basque) |
| be-BY | Belarusian (Belarus) |
| bg-BG | Bulgarian (Bulgaria) |
| ca-ES | Catalan (Catalan) |
| zh-HK | Chinese (Hong Kong SAR, PRC) |
| zh-MO | Chinese (Macao SAR) |
| zh-CN | Chinese (PRC) |
| zh-Hans | Chinese (Simplified) |
| zh-SG | Chinese (Singapore) |
| zh-TW | Chinese (Taiwan) |
| zh-Hant | Chinese (Traditional) |
| hr-HR | Croatian (Croatia) |
| cs-CZ | Czech (Czech Republic) |
| da-DK | Danish (Denmark) |
| dv-MV | Divehi (Maldives) |
| nl-BE | Dutch (Belgium) |
| nl-NL | Dutch (Netherlands) |
| en-AU | English (Australia) |
| en-BZ | English (Belize) |
| en-CA | English (Canada) |
| en-029 | English (Caribbean) |
| en-IE | English (Ireland) |
| en-JM | English (Jamaica) |
| en-NZ | English (New Zealand) |
| en-PH | English (Philippines) |
| en-ZA | English (South Africa |
| en-TT | English (Trinidad and Tobago) |
| en-GB | English (United Kingdom) |
| en-US | English (United States) |
| en-ZW | English (Zimbabwe) |
| et-EE | Estonian (Estonia) |

| Culture and language code | Culture |
|---|---|
| fo-FO | Faroese (Faroe Islands) |
| fa-IR | Farsi (Iran) |
| fi-FI | Finnish (Finland) |
| fr-BE | French (Belgium) |
| fr-CA | French (Canada) |
| fr-FR | French (France) |
| fr-LU | French (Luxembourg) |
| fr-MC | French (Monaco) |
| fr-CH | French (Switzerland) |
| gl-ES | Galician (Spain) |
| ka-GE | Georgian (Georgia) |
| de-AT | German (Austria) |
| de-DE | German (Germany) |
| de-LI | German (Liechtenstein) |
| de-LU | German (Luxembourg) |
| de-CH | German (Switzerland) |
| el-GR | Greek (Greece) |
| gu-IN | Gujarati (India) |
| he-IL | Hebrew (Israel) |
| hi-IN | Hindi (India) |
| hu-HU | Hungarian (Hungary) |
| is-IS | Icelandic (Iceland) |
| id-ID | Indonesian (Indonesia) |
| it-IT | Italian (Italy) |
| it-CH | Italian (Switzerland) |
| ja-JP | Japanese (Japan) |
| kn-IN | Kannada (India) |
| kk-KZ | Kazakh (Kazakhstan) |
| kok-IN | Konkani (India) |
| ko-KR | Korean (Korea) |
| ky-KG | Kyrgyz (Kyrgyzstan) |
| lv-LV | Latvian (Latvia) |
| lt-LT | Lithuanian (Lithuania) |

| Culture and language code | Culture |
|---|---|
| mk-MK | Macedonian (Macedonia, FYROM) |
| ms-BN | Malay (Brunei Darussalam) |
| ms-MY | Malay (Malaysia) |
| mr-IN | Marathi (India) |
| mn-MN | Mongolian (Mongolia) |
| nb-NO | Norwegian (Bokmål, Norway) |
| nn-NO | Norwegian (Nynorsk, Norway) |
| pl-PL | Polish (Poland) |
| pt-BR | Portuguese (Brazil) |
| pt-PT | Portuguese (Portugal) |
| pa-IN | Punjabi (India) |
| ro-RO | Romanian (Romania) |
| ru-RU | Russian (Russia) |
| sa-IN | Sanskrit (India) |
| sr-Cyrl-CS | Serbian (Serbia, Cyrillic) |
| sr-Latn-CS | Serbian (Serbia, Latin) |
| sk-SK | Slovak (Slovakia) |
| sl-SI | Slovenian (Slovenia) |
| es-AR | Spanish (Argentina) |
| es-BO | Spanish (Bolivia) |
| es-CL | Spanish (Chile) |
| es-CO | Spanish (Colombia) |
| es-CR | Spanish (Costa Rica) |
| es-DO | Spanish (Dominican Republic) |
| es-EC | Spanish (Ecuador) |
| es-SV | Spanish (El Salvador) |
| es-GT | Spanish (Guatemala) |
| es-HN | Spanish (Honduras) |
| es-MX | Spanish (Mexico) |
| es-NI | Spanish (Nicaragua) |
| es-PA | Spanish (Panama) |
| es-PY | Spanish (Paraguay) |
| es-PE | Spanish (Peru) |

| Culture and language code | Culture |
|---|---|
| es-PR | Spanish (Puerto Rico) |
| es-ES | Spanish (Spain) |
| es-ES_tradnl | Spanish (Spain, Traditional Sort) |
| es-UY | Spanish (Uruguay) |
| es-VE | Spanish (Venezuela) |
| sw-KE | Swahili (Kenya) |
| sv-FI | Swedish (Finland) |
| sv-SE | Swedish (Sweden) |
| syr-SY | Syriac (Syria) |
| ta-IN | Tamil (India) |
| tt-RU | Tatar (Russia) |
| te-IN | Telugu (India) |
| th-TH | Thai (Thailand) |
| tr-TR | Turkish (Turkey) |
| uk-UA | Ukrainian (Ukraine) |
| ur-PK | Urdu (Pakistan) |
| uz-Cyrl-UZ | Uzbek (Uzbekistan, Cyrillic) |
| uz-Latn-UZ | Uzbek (Uzbekistan, Latin) |
| vi-VN | Vietnamese (Vietnam) |

# Versions

## Versions of study components

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

To create a new version of a study component, install a Central Designer Incremental Deployment package, or submit a file containing the new component definition to the MedML Installer utility.

**Note:** Oracle recommends that you version study object definitions by installing a Central Designer Incremental Deployment package.

The MedML Installer utility creates a revised component definition in the study database for most component types, and the InForm application automatically uses the most recent component definitions when rendering forms in the browser, with the following exceptions:

- **PFElement**—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.

  For more information, see *PFElement* (on page 152).

- **StudyVersion**—Revisions provide the mechanism for updating a study version, and you must control the version numbers manually, by using the **VERSIONDESCRIPTION** attribute in the StudyVersion definition.

  For more information, see *StudyVersion* (on page 203).

- **StudyVersionSite**—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.

  For more information, see *StudyVersionSite* (on page 200).

- **StudyVersionDoc**—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.

  For more information, see *StudyVersionDoc* (on page 200).

**Note:** When you revise a study component definition, you cannot change its data type, which is specified in the DataType property.

## Versions of CRFs

When a CRF changes after a study has begun, sites use the new version of the CRF for as many study subjects as makes sense. For example, if a time-sensitive item such as a blood draw that must occur at a specified time is added to a form, the data will be collected for as many subjects as possible. In order to capture this data, a site requires the new version of the form. 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 study version.

- If a CRF change **does not** require sites to collect data from subjects for whom the form has already been completed:
  - Revise the Form component, including additional or changed controls, items, or sections, as appropriate.
- If a CRF change **does** require sites to collect data from subjects for whom the form has already been completed:

  1 Revise the Form component, including additional or changed controls, items, or sections, as appropriate.

  2 In the Form definition, include the **ALTREFNAME** attribute to indicate that you are creating an alternate definition of an existing form.

  When the InForm application runs, it presents the alternate version of the form along with the original version.

To implement a revised CRF, create a new StudyVersion that includes the form, and update the StudyVersionSite component for each site where the change applies.

For more information, see *Form* (on page 120), *StudyVersion* (on page 203), and *StudyVersionSite* (on page 200).

## Versions of the study protocol

The InForm application supports revisions to a study protocol and allows for the timing differences between when a protocol changes and when an Institutional Review Board (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 study version, including the new or changed forms, by copying or revising the definition of the old study version and updating the **VERSIONDESCRIPTION** attribute in the definition.

   For more information, see *StudyVersion* (on page 203).

2 Deploy the new StudyVersion definition at the sites by updating the **VERSIONDESCRIPTION** attribute of each StudyVersionSite 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).

   For more information, see *StudyVersionSite* (on page 200).

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

## Versions of documents

The InForm application supports revisions to all types of documents, including:

- Study 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 definition.

3 Load the new or updated document files into the database by using the MedML Installer utility to process the Documentation 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 to which the changed help text applies. If you updated CRF help using the Central Designer application, run the Incremental Deployment package that reflects the change.

5 Update the **VERSIONDESCRIPTION** attribute of the study version with which each document is associated.

6 Update the **VERSIONDESCRIPTION** attribute of the StudyVersionDoc definition to match the **VERSIONDEFINITION** attribute 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 component.

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

# The MedML Installer utility

## Overview of the MedML Installer utility

The MedML Installer utility:

- Parses the XML files that you generate with MedML elements.
- Validates the files against the following:
  - The MedML schema file (MedMLSchema.xsd).
  - The constraints enforced by the database schema.
- Loads the study components that the files define into the database.

You can run the MedML Installer utility:

- From the command line.

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

## Setting up a study with the MedML Installer utility

After all study components have been defined, use the MedML Installer utility to load the study-specific definitions for components in your study. These definitions are found in a set of XML files.

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

1 Create a set of XML files that define all the components of a study, using the MedML schema or the Central Designer application.

> **Note:** Oracle recommends that you design your study using the Central Designer application.

2 Process the XML files through the MedML Installer utility.

# Validation checks

## 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 study. 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, you can load definitions that are not fully compliant with the MedML specification for study design purposes only. Once the study is complete, use strict mode to ensure that the definitions you load are fully compliant with the MedML specification.

- 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 study, study 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

- Select **Start** > **Programs** > **Oracle** > **InForm 6.0** > **InForm MedML Installer**.

  OR

- Run the utility from the command line.

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

## About the MedML Installer window

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.

| Field | Description |
|---|---|
| XML File | The name of the file to add to (or modify in) the install. |
| XML Files to be Processed | A list of all of the files to be included in the next install. |
| Trial Name | The name of the study 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 | Creates a detailed description of what happened during the build. |
| Parse Only | Checks the input file for XML errors and compatibility with the InForm database without loading data. |
| Strict Mode | Accepts only complete component definitions.<br><br>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.<br><br>**Note:** This option is available only if you start the utility from the command line with the -notstrict option. Nonstrict mode is intended only for a study development environment. Do not use it in production.<br><br>Similarly, once a connection is defined, you can only load study definition data with strict MedML processing. |
| Online | Starts the study in online mode. By default, the checkbox is selected. In online mode, if a study is not started when you start the MedML Installer utility, the utility starts the study and does not shut it down when the installation is complete. This mode allows users to view the results of installing study definition data immediately.<br><br>If you deselect this checkbox, the MedML Installer utility starts the study in offline mode. In offline mode, the MedML Installer utility stops the study when installation is complete. Users cannot connect to the study until it is restarted.<br><br>**Note:** This option is available only if you start the utility from the command line with the -online option. |
| Output file | Location of a file that contains processing messages. |

# Working with the MedML Installer utility

## Running the MedML Installer utility for the first time

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

To run the MedML Installer utility:

1 Launch the MedML Installer utility. For more information, see *Launching the MedML Installer utility* (on page 36).

The MedML Installer window appears.

2 In the **XML File** field, select the XML file to process, or browse to 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 or type the name of the study.

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 36).

7 Click **Process**.

The MedML Installer utility processes the XML files and loads information into the study 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 *MedML Installer output messages* (on page 42).

## Updating a study that is already in progress

When you change a file in a study that is already in progress, you must process the new version of the file, and process a new study version to capture a new version of all of the objects that they changed file is referenced by. For example, if you make a change to a section, you must reprocess the file for the section and all files that contain references to that section.

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

To update a study that is already in progress:

1 Limit access to the study by changing the Directory Security properties of the study's virtual directory in **Internet Information Server** through **Microsoft Windows Components**.

For more information, see the *Study and Reporting Setup 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 or type the name of the study.

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 36).

7 Click **Process**.

The MedML Installer utility processes the XML files and loads information into the study 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 *MedML Installer output messages* (on page 42).

8 Stop and restart the study.

## Removing XML files from the build

Regulatory authorities have strict regulations against removing data from a study. XML files that are part of a study cannot be removed. You can instead remove all references to any unnecessary files in the other XML files, and reprocess all the files using the MedML Installer utility. The original file definitions for the removed files remain in the database, but are not visible in the study.

# Running the MedML Installer utility from the command line

**Note:** Oracle recommends that you run the MedML Installer utility through the PFConsole utility. For more information, see ***Running the PFConsole utility from the command line*** (on page 3).

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

```
PFConsole PFMMinst -trial trialname [-verbose] [-?] [-help] [-autorun] [-
nostrict]
[-outfile output file name] [-parse] -xml [@] XMLFile1 [@] XMLFile2 ...
[@] XMLFileN
```

**Note:** All command line applications, such as the MedML Installer utility, the InForm Data Import utility, and the InForm Data Export utility use the default product locale specified during the InForm installation, or through PFAdmin commands.

## Command line parameters

| Parameter | Variable | Description |
|---|---|---|
| PFConsole utility | | Starts the PFConsole utility. |
| PFMMinst | | Starts the MedML Installer utility. |
| -trial | trialname | The name of the study 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. |
| -autorun | | Runs the MedML Installer utility in a command window. Oracle recommends that you use the PFConsole utility to run the MedML Installer utility from the command line. For more information, see *PFConsole utility* (on page 1). |

| Parameter | Variable | Description |
|---|---|---|
| -notstrict | | Opens the MedML Installer box, which has a checkbox for specifying strict mode. By default the checkbox is deselected, 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.<br><br>**Note:** Use nonstrict mode only in a study development environment. Never load production study definitions in nonstrict mode. |
| -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. 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 with the list of XML files to be processed. |

## MedML Installer output messages

The MedML Installer utility generates three types of output messages during the installation process.

| Message | Description | Example |
|---|---|---|
| Information | Conditions that you should be aware of, but that probably will not cause serious problems with the file installation. | Information: Inserting PFELEMENT 'mestrUnk' |
| Warning | Events that might be detrimental and should be fixed, but that do not stop the file installation. | Warning: File "D:\PF408\FLTELT.xml' does not contain any known tags |
| Failure (Error) | An event that stops the file installation. | Error: Could not obtain child section 'sctIEInclusion'.<br><br>Error: Item not found |

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.

**Note:** The MedML Installer utility shows both errors and warnings in the Output Window and in the log file. You must correct errors before you can install the XML file.

# External Data Mapping

## Defining mappings to external data formats

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

- Clintrial database.
- Customer-Defined Database (CDD).

To load mapping definitions into the study database:

1 Generate mapping definitions as XML files.

2 Process the XML files with the MedML Installer utility.

## External mapping targets

The external entities for which you can generate mappings are:

- **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 (verbatims).
  - Controls in the InForm application to be used as code targets and context items.
- **Clintrial**—Mappings specify target items within panels in a Clintrial study definition.
- **Customer-Defined Database (CDD)**—Mappings specify target table columns within an external Oracle database.

# External Data Mapping Reference

## AssocCDD

### Purpose

Specifies the mapping in a customer-defined database of an association between two forms. An association definition is specified as a formset with **TYPE**=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 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. The options are true or false. True is the default. Optional.

**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 that defines the association. Required.

**LABEL**="*name*"

Text label for the association. The label can have a maximum of 255 characters. 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>

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

</EXTERNALMAP>
```

# CDD

## Purpose

Specifies the target of one mapped control in a CDD.

## 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|PATIENTTOSEC
TION|
    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 component is active. The options are true or false. True is the default. Optional.

**TARGETTABLE**="*name*"

Name of the target customer data table in which the mapped form item will be updated. The name must be 30 characters or fewer. Required.

**TARGETCOLUMN**="*name*"

Name of the column in the target customer data table in which the mapped form item will be added or updated. The name must be 25 characters or fewer. Required.

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

Type of data contained in the target column. Required.

The options are:

- **NUMERIC**

- **FLOAT**
- **DATE**— Data for a complete date and time field. Note that when you map a date and time field, the InForm software creates four columns:
  - **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**—Fewer than 255 characters.
- **TEXT**—Long varchar data.

**Note:** The **TARGETCOLUMNTYPE** attribute of a column that receives data from a multiple-selection control such as a checkbox control, must be set to STRING.

**TARGETCOLUMNMAXLENGTH**="*length*"

Maximum length of a target column with a **TARGETCOLUMNTYPE**=STRING. The column must be fewer than 255 characters. 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 submitted primary key, the InForm software 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, which 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*"

Indicates whether the column specified in the **TARGETCOLUMN** attribute is a pivot column. The options are true or false. 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, allowing access to an item in a target table with the PATIENTTOCONTROL or any of the PIVOT key types. The label must be 255 characters or fewer. Optional.

## Example

The following example illustrates the use of the CDD element to map the DESCRIBETEXT control to a column in the CDD1 CDD.

```
<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

Identifies the RefName of a formset as the location of a mapped control in the Path element of a mapping definition. Include one Chapterref element in a RefName path defined by a Path element.

### 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 element 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

Specifies the mapping of a control holding coded data in an InForm study to a specific type of code target in the dictionary used to code the data. The CodeTarget element is a child element in the CodingMap structure for a verbatim specified by the Path element 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

element 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 elements 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 element identifies the mappings for one verbatim in an InForm study, specified with the Path element, to be coded with the Central Coding application.

```
<EXTERNALMAP xmlns="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.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>
```

## CodingMap

### Purpose

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 WHO-DD dictionary.

### Children

A CodingMap element has the following child elements:

- One Dictionary element specifying the coding dictionary to use.
- At least one CodeTarget element 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 element containing one or more ContextItem elements. A ContextItem element 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

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 element has one or more ContextItem 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.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>
```

## ContextItem (mappings)

### Purpose

Specifies the mapping of a control holding coded data in an InForm study to a specific type of context item in the dictionary used to code the data. The ContextItem element is a child element in the ContextInformation structure for a verbatim specified by the Path element in a set of Central Coding 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 element 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 elements 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

```
<EXTERNALMAP xmlns="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="ATCCODE"
     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>
```

## ContextPanel

### Purpose

Defines a Clintrial panel in a set of mappings to be exported from the InForm database to the CIS application. A panel definition consists of a ContextPanel element, which is associated with CTItem elements defining the items common to all panels and one or more CTPanel elements, each associated with a single CTItem element defining a panel item.

### Syntax

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

### Attributes

**REFNAME**="*name*"

RefName of the set of Clintrial mappings. Required.

**ACTIVE**="*true*|*false*"

Indicates whether the component is active. The options are true or false. True is the default. 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.

### Children

The definition of a ContextPanel element can include one or more CTItem elements, 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 CTItem elements, each of which defines a panel item common to all panels.

```
<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>
```

## ControlRef

### Purpose

Includes 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:

- CheckBoxControl
- GroupControl
- Item
- RadioControl
- Path

Types of components you can include by using ControlRefs:

- CalculatedControl
- CheckBoxControl
- DateTimeControl
- GroupControl
- PullDownControl
- RadioControl
- SimpleControl
- TextControl

### 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 checkbox or radio control component labeled *Other* that is associated with a text box control.

**Note:** When defining compound controls with Controlref or Unitref 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 element, 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 drop-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 drop-down list as a set of PFElements, and include them in a PullDownControl definition by using Elementrefs.

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.

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

5 Include the pull-down control and text control in a RadioControl 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 element to identify the source control used in creating a mapping of components between the InForm database and a CDD.

```
<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

Defines the mapping of an InForm form item to a Clintrial panel item. The attributes of CTItem describe the attributes of the item in the 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"]
  [RANGELB="n"]
  [RANGEUB="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. The options are:

- Text
- Fixed
- Float
- Date
- 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. The options are:

- 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. The options are true or false. False is the default. Optional.

**ISREQUIRED**="*true*|*false*"

Indicates whether the item is required. The options are true or false. Required.

**DBFORMAT**="*format*"

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

Required.

**SASNAME**="*name*"

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

**CONTEXTTYPE**="*n*"

Context type of the item. 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. Types are:

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

Required.

**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. **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. Optional.

**CHECKLIST**="*text*"

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

**RANGELB**="*n*"

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

**RANGEUD**="*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. 0 (not a key item) is the default. Optional.

**COPYWITHPANEL**="*true*|*false*"

Indicates whether the item should be included with the panel if the panel is copied. Options are true or false. True is the default. Optional.

**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.

The default is 0. Optional.

**ISKEY**="*true*|*false*"

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

### 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>
```

# CTItem

## Purpose

Defines the mapping of an InForm form item to a Clintrial panel item. The attributes of CTItem describe the attributes of the item in the 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"]
  [RANGELB="n"]
  [RANGEUB="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. The options are:

- Text
- Fixed
- Float
- Date
- 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. The options are:

- 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. The options are true or false. False is the default. Optional.

**ISREQUIRED**="*true*|*false*"

Indicates whether the item is required. The options are true or false. Required.

**DBFORMAT**="*format*"

Format in which the Clintrial software stores values for the item in the Oracle database. The Oracle database formats are:

- VARCHAR2(*n*)
- DATE
- NUMBER(*xx*)
- NUMBER(*xx*,*yy*)
- NUMBER(*xx*,0).x

Required.

**SASNAME**="*name*"

Name of the item when data is sent to SAS through the Clintrial SAS interface. The name must be

eight characters or fewer and conform to SAS naming requirements. Optional.

**CONTEXTTYPE**="*n*"

Context type of the item. 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. Types are:

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

Required.

**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. **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. Optional.

**CHECKLIST**="*text*"

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

**RANGELB**="*n*"

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

**RANGEUD**="*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. 0 (not a key item) is the default. Optional.

**COPYWITHPANEL**="*true*|*false*"

Indicates whether the item should be included with the panel if the panel is copied. Options are true or false. True is the default. Optional.

**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.

The default is 0. Optional.

**ISKEY**="*true*|*false*"

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

### 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

Defines a Clintrial panel in a set of mappings to be exported from the InForm database to the CIS application. A panel definition consists of one or more CTPanel elements, each associated with a single CTItem element defining a panel item, and a ContextPanel element, which is associated with CTItem elements 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"]
  [MASTERCTITEM="name"]
  [SASNAME="name"]
  [LOCKSTATUS="n"]
  [ACTIVE="true|false"]
  [DESIGNNOTE="text"]>
  <CTITEM attributes/>
</CONTEXTPANEL>
```

### Attributes

**REFNAME**="*name*"

RefName of the set of Clintrial mappings. Required.

**PANELNAME**="*name*"

Name of the Clintrial panel. Required.

**DESCRIPTION**="*text*"

Description of the Clintrial panel. Optional.

**PANELTYPE**="*n*"

Clintrial panel type:

- 0—Non-subject data; that is, data is not related to a specific subject.
- 1—One record per subject; one record can be collected only one time during the study for each subject.
- 2—More than one record per subject; multiple records can be collected one time in the study for each subject.
- 3—One record per subject visit; one record can be collected for each subject visit.

- 4—More than one record per subject visit; multiple records can be collected for each subject visit.
- 5—One record for each enrolled subject.

Required.

**SUBSETITEM**="*name*"

Name of the item specified as the subset key for subset page sections based on the panel. A subset page section can occur multiple times on a study 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. Optional.

**ISPROTECTED**="*true*|*false*"

Indicates whether access rights to the panel are limited in Clintrial. The options are true or false. False is the default. Optional.

**ISVERIFIABLE**="*true*|*false*"

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

**ISDETAILPANEL**="*true*|*false*"

Indicates whether the CTPanel definition participates in a detail page section in a master-detail relationship: true or false. A master-detail relationship is a relationship between two page sections on a study 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. Required.

**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. This attribute is valid only if the value of the **ISDETAILPANEL** attribute is true. Optional.

**MASTERCTITEM**="*name*"

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

**SASNAME**="*name*"

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

**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.

The default is 0. Optional.

**ACTIVE**="*true*|*false*"

Indicates whether the component is active. The options are true or false. True is the default. 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.

### Children

The definition of a CTPanel element includes one CTItem element, 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 element maps this panel to a RefName path in an InForm study 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

Identifies a coding dictionary within a CodingMap 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 WHO-DD. Required.

**VERSION**="*text*"

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

**CULTURE**="*text*"

Language and culture, for example, en-JP. 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

A wrapper for the mappings for one control in an InForm study. The ExternalMap element can include:

- One Path element, which identifies the source control in the InForm study or signals the presence of mapping targets that have no corresponding control in the InForm study. For example:
  - Columns to be created in a CDD that have no associated control on a CRF.
  - Items that are common to all panels in a Clintrial protocol.
  - Multiple elements that specify the target of each mapping.

## Syntax

```
<EXTERNALMAP>
  <PATH/>
   <CDD* attributes/>
   <CONTEXTPANEL* attributes/>
   <CTPANEL* attributes/>
   <CODINGMAP* attributes/>
</EXTERNALMAP>
```

## Children

The ExternalMap element can include the following mapping definition elements:

- Path
- CDD
- ContextPanel
- CTPanel
- CodingMap

## Example

The following example illustrates the use of the ExternalMap element along with an unqualified Path element 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 study.

```
<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 element defines the mapping from a control in an InForm study, specified by the Path element, to a table column in a CDD, specified by the CDD element.

```
<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 element identifies the mappings for one verbatim in an InForm study, specified with the Path element, to be coded with the Central Coding application.

```
<EXTERNALMAP xmlns="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.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>
```

## ExternalMapSet

### Purpose

Identifies a set of external mappings with one RefName. Optional.

### Syntax

```
<EXTERNALMAPSET
  REFNAME="name"
  [DESIGNNOTE="text"]
  [ACTIVE="true|false"]
  TYPE="CDD|CLINFORM|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. The options are true or false. True is the default. Optional.

**TYPE**="CDD|CLINFORM|CODINGMAP"

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

- **CDD**—Table columns within an external Oracle database.
- **CLINFORM**—Items within panels in a Clintrial study definition.
- **CODINGMAP**—Central Coding application.

Required.

### Example

The following example illustrates the ExternalMapSet element in the MedML that defines the MEDIKA-CLINICAL sample study.

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

# Itemref

## Purpose

- Includes previously defined items in the definition of a section, itemset, or item group. Include one Itemref element for each item in the section, itemset, or item group definition.
- Specifies the RefName of an item as the location of a mapped control in the Path element of a mapping definition. Include one Itemref element in a RefName path defined by a Path element.

An itemref appears only as the child of a section or path 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 that the itemref is including in the definition of a section or path. Required.

**KEYITEM**="*true*|*false*"

Indicates whether the item that the itemref is including is a key item in an itemset definition. The values of key items are displayed in a drop-down list in the InForm application to allow you 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 checkbox controls, group controls, radio controls, or drop-down lists cannot be key items. Options are true or false. False is the default. Optional.

**Note:** You can specify key items for repeating forms by using the KeyItemref element. 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, form, and itemset. true or false. 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. Options are true or false. False is the default. Optional.

**ORDER**="*n*"

Sequence in which each itemref appears in the Section definition. 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. Optional.

**Note:** When including an Itemref definition in a Path element, only the **REFNAME** attribute is valid.

## Examples

This example illustrates the use of itemrefs in the definition of a section called Duration of Hypertension, which contains two items. For more information, see *Section* (on page 177).

```
<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 element. 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 elements.

```
<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

Identifies the RefName of an itemset as the location of a mapped control in the Path element of a mapping definition. Include one ItemSet element in a RefName path defined by a Path element.

### 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>
```

## Pageref

### Purpose

Identifies the RefName of a form as the location of a mapped control in the Path element of a mapping definition. Include one Pageref element in a RefName path defined by a Path element.

### 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

- Used alone, with no child components, a Path element signals the presence of mapping targets that have no corresponding control in the InForm study. For example:
  - Columns to be created in a CDD 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 element specifies a control in an InForm study. This control identifies the source data for one data mapping.

The Path element is a required component of a mapping definition created by the ExternalMap element and must be the first child component element in the ExternalMap definition.

### Syntax

```
<PATH>
 <CHAPTERREF attributes/>
  <PAGEREF attributes/>
  <SECTIONREF attributes/>
```

```
  <ITEMSETREF attributes/>
  <ITEMREF attributes/>
  <CONTROLREF* attributes/>
</PATH>
```

### Children

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

- Chapterref
- Pageref
- Sectionref
- Itemsetref
- Itemref
- Controlref

### Example

This example illustrates the use of the Path element 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>
```

# Sectionref

## Purpose

- Includes previously defined sections in the definition of a form.
- Includes one Sectionref element for each section in the form.
- Specifies the RefName of a section as the location of a mapped control in the Path element of a mapping definition.
- Includes one Sectionref element in a RefName path defined by a Path element.

A sectionref appears only as the child of a Form or Path element 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 that the Sectionref is including in the definition of a form or path. Required.

**ORDER**="*n*"

Sequence in which each sectionref appears in the form definition. 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. Optional.

**Note:** When including a Sectionref definition in a Path element, 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: Demographics (DEM) and Smoking History (SH).

```
<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

Specifies the action that results when an associated event 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. 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 in the Open or Candidate state. Optional.

**LANGUAGE**="*name*"

Code for the culture and language used to display text values. If you do not include a **LANGUAGE** attribute, the default is the installed default product locale. Required if text strings are translated (if a **TRANSLATIONS** element is included as a child of the study component definition).

**Note:** Relying on the default product locale is not recommended, because the default product locale can be changed.

**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 the Open state with the text "Value Out of Range." The action is included in the definition of an event that also executes an execution plan 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

Defines dependencies between a generic rule and an item, itemset, or form. The attributes and child components in an AttachRuleDepend group define a dependency for the context in which a rule runs.

A context can be specific to a particular visit, form, section, and itemset or item, or it can have a generic scope in either of the following ways:

- **Generic Visit context**—Associates a rule with a visit.
- **Generic Form context**—Associates a rule with a form/section/item combination in every visit 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 element.

**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 checkbox 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. The options are:

- **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.

**APPLIED** is the default. Optional.

# AttachRuleSet

## Purpose

Associates a rule with a visit, an item or an itemset on a form, or with an association between two related forms. An active rule that is associated with an item or itemset executes when the item or itemset is submitted after initial entry, or after the item or itemset is changed. A rule attached to an association executes when a user selects or deselects the checkbox 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 form the context in which a rule runs.

A context can be specific to a particular visit, form, section, and itemset or item, or it can have a generic scope in either of the following ways:

- **Generic Visit context**—Associates a rule with a visit.
- **Generic Form context**—Associates a rule with a form/section/item combination in every visit 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 element.

**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.

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. This name must be unique for each rule. Required.

**RULENAME**="*name*"

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

**FORMSETNAME**="*name*"

Name of the formset in which the item appears. 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. Required.

**FORMNAME**="*name*"

Name of the form in which the item appears. 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. Required.

**SECTIONNAME**="*name*"

Name of the section of the form in which the item appears. 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. Required.

**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. Help text specified in the AttachRuleSet element overrides help text specified in the rule. Optional.

**ATTACHTYPE**="APPLIED|DEPENDENCY|TRIGGER"

The type of dependency you want to use. The options are:

- **APPLIED**—Rule fires when the item or itemset is submitted after initial entry, or after the item or itemset is 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.

**APPLIED** is the default. Optional.

**ACTIVE**="*true*|*false*"

Indicates whether the component is active. The options are true or false. True is the default. Optional.

### Children

The definition of an AttachRuleSet can include the following:

- Rulearg elements, each of which defines the parameters of the rule.
- AttachRuleDepend elements. Each AttachRuleDepend element indicates the type of dependency and the item or the association the rule is dependent on.
- Eventref elements. Each Eventref element allows you to include the definition of an event in the definition of a rule. 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

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 reserved for use by Oracle.**

### Syntax

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

### Attributes

**BROWSERNAME**="*name*"

Name of the browser. Required.

**BROWSERTYPE**="*type*"

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

### Example

The following example illustrates the XML code used to define the browser component for Internet Explorer version 9.0.

```
<BROWSER BROWSERNAME="IE090" BROWSERTYPE="16897"/>
```

# CalculatedControl

### Purpose

Defines 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">
      <UNITREF attributes/>
      <TRANSLATIONS/>
</CALCULATEDCONTROL>
```

### Attributes

**REFNAME**="*name*"

Name used when referring to the calculated control in the definition of another control. Required. This name must be unique among calculated controls.

**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 calculated control in the definition of another control. Optional.

**UUID**="*id*"

Universally Unique Identifier; a string that identifies the component uniquely across all studies, study databases, and machines. Optional.

**CAPTION**="*text*"

Text that appears with the calculated value. You can use the **TRANSLATIONS** attribute to provide translations of the attribute value. Optional.

**LANGUAGE**="*name*"

Code for the culture and language used to display text values. If you do not include a **LANGUAGE** attribute, the default is the installed default product locale. Required if text strings are translated (if a **TRANSLATIONS** element is included as a child of the study component definition).

**Note:** Relying on the default product locale is not recommended, because the default product locale can be changed.

**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. Left is the default. Optional.

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

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

Left is the default. Optional.

**FORMAT**="*return_format*"

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

**UNITDISPLAYTYPE**="ELEMENT"

Type of control used to display units included in the text control with a Unitref 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 dependents for the calculation. If you remove only the calculated item itself, the rule system fails.

## Children

- Zero or one Unitref element that specifies the unit to be associated with the control.
- Zero or one Translations definition that specifies a set of translated strings for the translatable attributes of the study component. A Translations element must include one or more Translation definitions. Each Translation element represents the translation of one attribute of a study component into the language of one locale. The Translations element must be the last child element in the study component definition. In a CalculatedControl definition, you can translate the **CAPTION** attribute.

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

Additionally, the example illustrates the definition of a translated string for the calculated control caption.

```
<CALCULATEDCONTROL REFNAME="RANDOMIZATION"
   UUID="DC2EB0BF-4F12-11d2-9319-00A0C9769A13"
   LANGUAGE="en-US"
   CAPTION="Drug Kit"
   NAME="RANDOMIZATION"
   FORMAT="Drug-kit number %s has been assigned to this patient">
      <TRANSLATIONS>
         <TRANSLATION NAME="CAPTION" LOCALE="fr-FR"
         DISPLAYTEXT="Kit de Drogues"/>
      </TRANSLATIONS>
</CALCULATEDCONTROL>
<ITEM REFNAME="DRUGKIT"
   UUID="52AF1207-4F13-11d2-9319-00A0C9769A13"
   QUESTION="Sequence Number and Information: "
   ITEMREQUIRED="false"
   CALCULATED="true">
   <CONTROLREF REFNAME="RANDOMIZATION" />
```

</ITEM>

# CheckboxControl

## Purpose

Defines a list of checkboxes from which users can select one or more options. Checkbox controls are composed of previously defined components that you include by using Controlrefs. You can include the following types of components in a CheckBoxControl definition:

- CalculatedControl
- CheckBoxControl
- GroupControl
- PullDownControl
- RadioControl
- SimpleControl
- TextControl

## 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/>
      <TRANSLATIONS/>
</CHECKBOXCONTROL>
```

### Attributes

**REFNAME**="*name*"

Name used when referring to the checkbox control in the definition of another control. Required. This name must be unique among checkbox controls.

**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 checkbox control in the definition of another control. Optional.

**UUID**="*id*"

Universally Unique Identifier; a string that identifies the component uniquely across all studies, study databases, and machines. Optional.

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

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

**LAYOUT**="VERTICAL|HORIZONTAL|NOWRAP"

Orientation of the checkboxes: Vertical, Horizontal, or Nowrap. 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. Optional.

**CAPTION**="*text*"

Text that appears on the screen with the checkbox list. You can use the **TRANSLATIONS** attribute to provide translations of the attribute value. Optional.

**LANGUAGE**="*name*"

Code for the culture and language used to display text values. If you do not include a **LANGUAGE** attribute, the default is the installed default product locale. Required if text strings are translated (if a **TRANSLATIONS** element is included as a child of the study component definition).

**Note:** Relying on the default product locale is not recommended, because the default product locale can be changed.

**CAPTIONALIGN**="LEFT|RIGHT|TOP|BOTTOM"

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

## Children

- One or more Controlref definitions. Each Controlref refers to a previously defined component that identifies one item in the list of checkboxes.

  **Note: When defining compound controls with Controlref 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.**

- Zero or one Translations definition that specifies a set of translated strings for the translatable attributes of the study component. A Translations element must include one or more Translation definitions. Each Translation element represents the translation of one attribute of a study component into the language of one locale. The Translations element must be the last child element in the study component definition. In a CheckboxControl definition, you can translate the **CAPTION** attribute.

## Example 1

This example illustrates how to create a simple checkbox control by including previously defined PFElements that have been enclosed in simple controls. The checkbox control in this example is a list of subject smoking habits.

1 Define each check box component as a PFElement, and enclose it in a simple control. 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>
<SIMPLECONTROL REFNAME="ND" NAME="ND">
   <ELEMENTREF REFNAME="ND"/>
</SIMPLECONTROL>
```

3 Define a checkbox control that includes each simple control with a Controlref.

4
```
<CHECKBOXCONTROL REFNAME="SMOKING"
  NAME="SMOKING"
  LAYOUT="VERTICAL">
  <CONTROLREF REFNAME="CIGARETTE" ORDER="1"/>
  <CONTROLREF REFNAME="PIPE" ORDER="2"/>
  <CONTROLREF REFNAME="CIGAR" ORDER="3"/>
  <CONTROLREF REFNAME="ND" ORDER="4"/>
</CHECKBOXCONTROL>
```

## Example 2

This example illustrates a checkbox control with nested components. The sample checkbox control is for collecting data about muscle aches.

1 Define text controls for the Left, Right, and Other text boxes. Note that the LEFT text control uses the HTML special character   to add spaces that control the alignment with the RIGHT text control.

2
```
<TEXTCONTROL REFNAME="LEFT" NAME="LEFT"
  CAPTION="Left:&amp;nbsp;&amp;nbsp;" CAPTIONALIGN="LEFT"
  HEIGHT="1"
  LENGTH="32"
  MAXLENGTH="50"
  DATATYPE="STRING"/>
<TEXTCONTROL REFNAME="RIGHT" NAME="RIGHT"
  CAPTION="Right :" CAPTIONALIGN="LEFT"
  HEIGHT="1"
  LENGTH="32"
  MAXLENGTH="50"
  DATATYPE="STRING"/>
<TEXTCONTROL REFNAME="OTHER" NAME="OTHER"
  CAPTION="Other (describe):" CAPTIONALIGN="TOP"
  HEIGHT="1"
  LENGTH="32"
  MAXLENGTH="50"
  DATATYPE="STRING"/>
```

3 Define CheckBoxControls for the Arms and Legs check boxes by using Controlrefs. Note that you can use the Left and Right text controls in each checkbox control.

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 checkbox control that uses Controlrefs to include the Arms and Legs checkbox controls and the Other text control.

6
```
<CHECKBOXCONTROL REFNAME="MUSCLES" NAME="MUSCLES"
  LAYOUT="VERTICAL">
    <CONTROLREF REFNAME="ARMS" ORDER="1"/>
    <CONTROLREF REFNAME="LEGS" ORDER="2"/>
    <CONTROLREF REFNAME="OTHER" ORDER="3"/>
</CHECKBOXCONTROL>
```

### Example 3

The following example illustrates the definition of a translation for a checkbox control caption.

```
<CHECKBOXCONTROL REFNAME="MUSCLES"
   NAME="MUSCLES"
   LANGUAGE="en-US"
   CAPTION="Principal location of pain"
   LAYOUT="VERTICAL">
      <CONTROLREF REFNAME="ARMS" ORDER="1"/>
      <CONTROLREF REFNAME="LEGS" ORDER="2"/>
      <CONTROLREF REFNAME="OTHER" ORDER="3"/>
      <TRANSLATIONS>
         <TRANSLATION NAME="CAPTION" LOCALE="es-ES"
         DISPLAYTEXT="Principal Localización del Dolor"/>
         <TRANSLATION NAME="CAPTION" LOCALE="it-IT"
         DISPLAYTEXT="Principali Località del Dolore"/>
      </TRANSLATIONS>
</CHECKBOXCONTROL>
```

## CodeTarget (dictionary)

### Purpose

Part of the element 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

Part of the element 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 item, such as:

- **Route of Administration**—The route by which the drug was administered (WHO-DD only).
- **Indication**—The disease or disorder for which the drug was taken (WHO-DD only).

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>
```

# Controlref

## Purpose

Includes 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:

- CheckBoxControl
- GroupControl
- Item
- RadioControl
- Path

Types of components you can include by using Controlrefs:

- CalculatedControl
- CheckBoxControl
- DateTimeControl
- GroupControl
- PullDownControl
- RadioControl
- SimpleControl
- TextControl

## 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 checkbox or radio control component labeled *Other* that is associated with a text box control.

**Note:** When defining compound controls with Controlref or Unitref 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 element, 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 drop-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 drop-down list as a set of PFElements, and include them in a PullDownControl definition by using Elementrefs.

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.

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

5 Include the pull-down control and text control in a RadioControl 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 element to identify the source control used in creating a mapping of components between the InForm database and a CDD.

```
<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

Defines a set of drop-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"]>
      <TRANSLATIONS/>
</DATETIMECONTROL>
```

### Attributes

**REFNAME**="*name*"

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

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

**UUID**="*id*"

Universally Unique Identifier; a string that identifies the component uniquely across all studies, study databases, and machines. Optional.

**Note:** When you create a repeating formset, 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*"

Code for the culture and language used to display text values. If you do not include a **LANGUAGE** attribute, the default is the installed default product locale. Required if text strings are translated (if a **TRANSLATIONS** element is included as a child of the study component definition).

**Note:** Relying on the default product locale is not recommended, because the default product locale can be changed.

**CAPTION**="*text*"

Text that appears on the screen with the checkbox list You can use the **TRANSLATIONS** attribute to provide translations of the attribute value. Optional.

**CAPTIONALIGN**="LEFT|RIGHT|TOP|BOTTOM"

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

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

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

**STARTYEAR**="*year*"

First year that appears in the drop-down list for the year. Required when you use the

DateTimeControl to define a date.

**ENDYEAR**="*year*"

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

**DISPLAYMONTH**="*true*|*false*"

Indicates whether the control includes a drop-down list for the month. True is the default. Optional.

**DISPLAYDAY**="*true*|*false*"

Indicates whether the control includes a drop-down list for the day. True is the default. Optional.

**DISPLAYYEAR**="*true*|*false*"

Indicates whether the control includes a drop-down list for the year. True is the default. Optional.

**DISPLAYHOUR**="*true*|*false*"

Indicates whether the control includes a drop-down list for the hour. False is the default. Optional.

**DISPLAYMINUTE**="*true*|*false*"

Indicates whether the control includes a drop-down list for the minute. False is the default. Optional.

**DISPLAYSECOND**="*true*|*false*"

Indicates whether the control includes a drop-down list for the second. False is the default. Optional.

**REQUIREMONTH**="*true*|*false*"

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

**REQUIREDAY**="*true*|*false*"

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

**REQUIREYEAR**="*true*|*false*"

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

**REQUIREHOUR**="*true*|*false*"

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

**REQUIREMINUTE**="*true*|*false*"

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

**REQUIRESECOND**="*true*|*false*"

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

**UNKNOWNMONTH**="*true*|*false*"

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

**UNKNOWNDAY**="*true*|*false*"

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

**UNKNOWNYEAR**="*true*|*false*"

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

**UNKNOWNHOUR**="*true*|*false*"

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

**UNKNOWNMINUTE**="*true*|*false*"

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

**UNKNOWNSECOND**="*true*|*false*"

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

**CHECKCONSISTENT**="*true*|*false*"

Specifies whether the InForm application checks for internal consistency among the entered components of the date and time. True is the default. Optional. 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 software 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 drop-down lists. False is the default. Optional.

### Children

Zero or one Translations definition that specifies a set of translated strings for the translatable attributes of the study component. A Translations element must include one or more Translation definitions. Each Translation element represents the translation of one attribute of a study component into the language of one locale. In a DateTimeControl definition, you can translate the **CAPTION** attribute.

### Example 1

The following example illustrates a DateTimeControl definition that creates drop-down lists for the subject birth month, day, and year. Month, day, and year are required. The control is included in an Item definition by using the Controlref component. Note that this example also includes a HelpLink component that specifies a link from the item to its CRF Help entry.

```
<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 definition that is required in the first form of every visit and in a repeating formset. This section definition uses a datetime control with a required UUID. The control includes the month, day, year, hour, and minute of the visit. Month and year are required.

```
<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>

### Example 3

This example illustrates the translation of a caption in a DateTimeControl definition.

```
<DATETIMECONTROL REFNAME="startdate" NAME="startdate"
   LANGUAGE="en-US"
   STARTYEAR="1990"
   ENDYEAR="2010"
   CAPTION="Enter Month/Day/Year"
   DISPLAYDAY="true" DISPLAYMONTH="true" DISPLAYYEAR="true"
   REQUIREMONTH="true" REQUIREDAY="true" REQUIREYEAR="true">
      <TRANSLATIONS>
         <TRANSLATION NAME="CAPTION" LOCALE="fr-FR"
         DISPLAYTEXT="Entrez Jour / Mois / Année"/>
      </TRANSLATIONS>
</DATETIMECONTROL>
```

## Dictionary (definition)

### Purpose

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 WHO-DD. Required.

**VERSION**="*text*"

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

**CULTURE**="*text*"

Language and culture; for example, en-JP. Required.

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

### Children

- One or more CodeTarget elements identifying a specific code target defined for the dictionary.
- Zero or more ContextItem elements identifying a specific context item defined for the dictionary.
- Zero or more VerbatimType elements identifying a valid verbatim type for the dictionary. The verbatim type corresponds to the Central Coding item type.

### Restrictions

- 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

Defines a single HTM file that typically represents a section of a study protocol, a set of CRF item help for a CRF, or an online help topic. DocBody elements are defined by inclusion in a Documentation definition; they do not stand alone.

## Syntax

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

## 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. You can use the **TRANSLATIONS** attribute to specify the path and file name of a file that contains a translated version of the content represented by the DocBody element.

**Note:** The path must be relative to the location from which you are running the MedML Installer utility.

**FILENAMEID**="*id*"

Identification code used only by synchronization processing.

**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. The default is the order in which the DocBody appears in the Documentation definition. Optional.

**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.

**LANGUAGE**="*name*"

Code for the culture and language used to display text values. If you do not include a **LANGUAGE** attribute, the default is the installed default product locale. Required if text strings are translated (if a **TRANSLATIONS** element is included as a child of the study component definition).

**Note:** Relying on the default product locale is not recommended, because the default product locale can be changed.

## Children

Zero or one Translations definition that specifies a set of translated files. A Translations element must include one or more Translation definitions. Each Translation element contains the path and file name of a file that is translated into the language of one locale. In a DocBody definition, you can include the path and file name of a translated HTM file that typically represents a section of a study protocol, a set of CRF item help for a CRF or an online help topic.

## Example

The following example illustrates the use of DocBody elements to load the topics in one section of online Help. Additionally, it shows how to load translated versions of the topics.

```
<DOCUMENTATION REFNAME="AboutInForm"
   DOCNAME="About InForm software"
   DOCTYPE="HELP"
   HELPTEXT="Click here for an introduction to InForm software"
   TOC="..\XMLBase\Help\TocAboutInform.htm"
   TOCLINKS="6"
   INDEX="..\XMLBase\Help\HelpIndex.htm"
   INDEXLINKS="220">
      <DOCBODY REFNAME="WELCOME" FILENAME="..\XMLBase\Help\Welcome.htm"
      LINKS="5"/>
         <TRANSLATIONS>
            <TRANSLATION NAME="FILENAME" LOCALE="jp-JP"
            DISPLAYTEXT="..\XMLBase\Help\Welcome_jp-JP.htm"/>
         </TRANSLATIONS>
      <DOCBODY REFNAME="FEATURES" FILENAME="..\XMLBase\Help\InFormFeatures.htm"
      LINKS="4"/>
         <TRANSLATIONS>
            <TRANSLATION NAME="FILENAME" LOCALE="jp-JP"
            DISPLAYTEXT="..\XMLBase\Help\InFormFeatures_jp-JP.htm"/>
         </TRANSLATIONS>
      <DOCBODY REFNAME="ROADMAP" FILENAME="..\XMLBase\Help\RoadMap.htm"
      LINKS="5"/>
```

```
         <TRANSLATIONS>
            <TRANSLATION NAME="FILENAME" LOCALE="jp-JP"
            DISPLAYTEXT="..\XMLBase\Help\RoadMap_jp-JP.htm"/>
         </TRANSLATIONS>
      <DOCBODY REFNAME="RIGHTS" FILENAME="..\XMLBase\Help\Rights.htm"
      LINKS="5"/>
         <TRANSLATIONS>
            <TRANSLATION NAME="FILENAME" LOCALE="jp-JP"
            DISPLAYTEXT="..\XMLBase\Help\Rights_jp-JP.htm"/>
         </TRANSLATIONS>
      <DOCBODY REFNAME="DATETIME" FILENAME="..\XMLBase\\Help\DateTime.htm"
      LINKS="4"/>
         <TRANSLATIONS>
            <TRANSLATION NAME="FILENAME" LOCALE="jp-JP"
            DISPLAYTEXT="..\XMLBase\Help\DateTime_jp-JP.htm"/>
         </TRANSLATIONS>
</DOCUMENTATION>
```

# Documentation

## Purpose

Defines 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 element. In the InForm application, the following entities are represented by the Documentation element:

- Study protocol
- CRF help
- Sponsor documents
- Visit Calculator
- Sample Case Book

For a document to be visible in the Document or Help window, the Documentation definition must be associated with a StudyVersion in the StudyVersionDoc element.

## 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/>
      <TRANSLATIONS/>
</DOCUMENTATION>
```

### Attributes

**REFNAME="name"**

Name used when referring to the document in a StudyVersionDoc definition. Required.

**DOCNAME**="*name*"

Name that appears on the element that represents the Document in the Document or Help window. You can use the **TRANSLATIONS** attribute to provide translations of the attribute value. Required.

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

Type of document. Required.

- **BOOKMARKDOC**—A document that other documents can link to; used for study protocol.
- **BOOKMARKFAQDOC**—A document that other documents can link to; used for CRF Help.
- **VISITDOC**—The Visit Calculator screen.
- **CRBDOC**—A blank, sample set of CRF forms for the study.

**HELPTEXT**="*text*"

Text that appears as hover help when a user moves the cursor over the document's tab in the Document or Help window. You can use the **TRANSLATIONS** attribute to provide translations of the attribute value. Optional.

**UUID**="*id*"

Universally Unique Identifier; a string that identifies the component uniquely across all studies, study databases, and machines. Optional.

**LANGUAGE**="*name*"

Code for the culture and language used to display text values. If you do not include a **LANGUAGE** attribute, the default is the installed default product locale. Required if text strings are translated (if a **TRANSLATIONS** element is included as a child of the study component definition).

**Note:** Relying on the default product locale is not recommended, because the default product locale can be changed.

**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. You can use the **TRANSLATIONS** attribute to specify the path and file name of a file that contains a translated table of contents.

**Note:** 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. You can use the **TRANSLATIONS** attribute to specify the path and file name of a file that contains a translated index.

**Note:** 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

- Zero or more DocBody elements. Each DocBody element defines a single HTM file that typically represents a section of a study 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 elements.
- Zero or one Translations definition that specifies a set of translated strings for the translatable attributes of the study component. A Translations element must include one or more Translation definitions. Each Translation element represents the translation of one attribute of a study component into the language of one locale. The Translations element must be the last child element in the study component definition. In a Documentation definition, you can translate the **DOCNAME** and **HELPTEXT** attributes, and you can specify the path and file name of a translated table of contents and index. If you translate any attributes, include translation definitions for all translatable attributes that are part of the study component definition.

### Example

The following set of Documentation definitions illustrates each document type. The Protocol, Study, and Help documents include multiple HTM files, defined with DocBody elements. The Protocol and Study documents have tables of contents. The Help document has both a table of contents and an index. Translations are provided for the DOCNAME, HELPTEXT, TOC, and INDEX attributes of the Documentation definition.

```
<DOCUMENTATION REFNAME="Protocol"
   DOCNAME="Protocol Guide"
   DOCTYPE="BOOKMARKDOC"
   LANGUAGE=”en-US”
   HELPTEXT="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"
   LANGUAGE=”en-US”
   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="SSHELP" FILENAME=".\StudyGuide\Signs_Symptoms.htm"/>
      <DOCBODY REFNAME="DEMHELP" FILENAME=".\StudyGuide\Demographics.htm"/>
      <DOCBODY REFNAME="VSHELP" 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"
   LANGUAGE=”en-US”
   HELPTEXT="Click here to view Patient Visit Calculator"
</DOCUMENTATION>


<DOCUMENTATION REFNAME="CRB"
   DOCNAME="Sample Book"
   DOCTYPE="CRBDOC"
   LANGUAGE=”en-US”
   HELPTEXT="Click here to view Sample Case Book forms"
</DOCUMENTATION>
```

# Elementref

## Purpose

Includes the definition of a PFElement in the definition of the following other types of form components:

- PullDownControl
- SimpleControl

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 that the Elementref is including in the PullDownControl or SimpleControl. Required.

**ORDER**="*n*"

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

## Example

This example demonstrates how to use Elementrefs to include three previously created PFElements that define selections describing a subject body frame in a PullDownControl called "FRAME." In the example, the list is ordered in the same sequence in which the entries appear; to reorder the list, 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

Defines an event that executes when its rule fires. The definition of an event may include a reference to an execution plan, and/or 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. This name must be unique among events. 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; a string that identifies the component uniquely across all studies, study databases, and machines. Required.

## Children

The definition of an event can include zero or more action elements, 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 elements. Each ExecutionPlanref refers to a previously defined execution plan. When an event executes, any included execution plans also execute.

## Example

The following example illustrates an event that generates a query in the Open state with the text "Value Out of Range." The event also executes an execution plan 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

Includes the definition of an event in the definition of a rule. 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 that the Eventref is including in the rule definition. Required.

## Example

In the following example, an Eventref element includes the Value Out of Range event definition in the definition of a rule 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

Defines a script that can do the following:

- Send e-mail.
- Log an entry in the Windows 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. This name must be unique among ExecutionPlans. 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.

**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 execution plan that logs a message to the Windows log when the event in which it is included fires.

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

# ExecutionPlanref

## Purpose

Includes the definition of an execution plan in the definition of an event. 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 execution plan that the ExecutionPlanref is including in the event definition. Required.

## Example

In this example, the ExecutionPlanref element includes the EPScript execution plan definition in the Value Out of Range event 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>
```

# Formref

## Purpose

Includes a previously defined form in the definition of a formset. 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 that the Formref is including in the definition of a formset. 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. The options are true or false. False is the default.

**ORDER**="*n*"

Sequence in which each form appears in the formset definition. 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. Optional.

**ALTREFNAME**="*name*"

Identifies a new version of an old form to be presented to a subject who is participating in the study when the study version changes. 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. Optional.

**VISITREFNAME**="*name*"

RefName of the formset in which an association between forms 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 elements group the Demographics (DEM) and Hypertension History (HH) forms into a formset that make up the first visit in the study, Week -4. For more details, see ***FormSet*** (on page 124).

```
<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 elements 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>
```

# Form

## Purpose

Creates the definition of a complete form. A form consists of a group of predefined sections, included by using Sectionref elements.

## Syntax

```
<FORM
   REFNAME="name"
   [DESIGNNOTE="text"]
   [UUID="id"]
   TITLE="text"
   MNEMONIC="name"
   [NOTE="text"]
   [LANGUAGE="name"]
   [TYPE="CRF|ENROLLMENT|REGDOC|VISITREPORT|CUSTOM|CUSTOMTRIAL|CUSTOMRULES|
          CUSTOMADMIN|CUSTOMAUTH"]
   [QUESTIONWIDTH="n"]
   [CONTROLWIDTH="n"]
   [REPEATING="true|false"]
   [UNIQUEKEY="true|false"]>
      <SECTIONREF+ attributes/>
      <HELPLINK attributes/>
      <KEYITEMREF attributes/>
      <TRANSLATIONS/>
</FORM>
```

## Attributes

**REFNAME**="*name*"

Name used when referring to the form in the definition of a formset. This name must be unique among forms. 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; a string that identifies the component uniquely across all studies, study databases, and machines. Required.

**TITLE**="*text*"

Text of the form title, displayed at the top of the form on the screen. You can use the

**TRANSLATIONS** attribute to provide translations of the attribute value. Required.

**MNEMONIC**="*name*"

Abbreviation of the form title, displayed on the tab control that a user clicks to navigate to a form. You can use the **TRANSLATIONS** attribute to provide translations of the attribute value. Required.

**NOTE**="*text*"

Text of a note displayed immediately below the form title on the screen. You can use the **TRANSLATIONS** attribute to provide translations of the attribute value. Optional.

**LANGUAGE**="*name*"

Code for the culture and language used to display text values. If you do not include a **LANGUAGE** attribute, the default is the installed default product locale. Required if text strings are translated (if a **TRANSLATIONS** element is included as a child of the study component definition).

**Note:** Relying on the default product locale is not recommended, because the default product locale can be changed.

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

Type of form. The options are:

- **CRF**—Case Report Form. If the CRF can have multiple instances within the same visit, use the REPEATING attribute.
- **ENROLLMENT**—Screening form or Enrollment form.
- **REGDOC**—Form used for the Regulatory Document page.
- **VISITREPORT**—Form used for the Visit Report page.
- **CUSTOM**—Not used.
- **CUSTOMTRIAL**—Form used for pages that are integral to InForm functionality.
- **CUSTOMRULES**—Form used for rules administration pages.
- **CUSTOMADMIN** — Form used for administration pages such as the User, Site, and System Configuration pages.
- **CUSTOMAUTH**—Form used for password administration pages.

CRF is the default. Optional.

**QUESTIONWIDTH**="*n*"

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

**CONTROLWIDTH**="*n*"

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

**REPEATING**="*true*|*false*"

Indicates whether the form is repeating. In a repeating form, multiple instances can occur within the same visit. Repeating forms can also be related in an association. The options are true or false. False

is the default. Optional.

**UNIQUEKEY**="*true*|*false*"

Indicates whether the form contains one or more composite keys; that is, items defined with group uniqueness in the study design. This property applies only to repeating forms. False is the default. Optional.

If the form contains a composite key and two instances of the form are submitted in the same visit with the same composite key items, the second instance is rejected. Items within itemsets cannot be repeating form key items.

If you specify true:

- The combination of all of the key items must be unique.
- The setting does not override the UNIQUEKEY attribute of the key items; they still need to be individually unique. As a result, it does not make sense to specify true for both the ITEMSET and ITEMREF elements.
- Repeating form unique keys are only checked across identical form versions.
- When checking repeating form unique keys, deleted repeating form instances are not considered.

  You cannot undelete a deleted repeating form instance if doing so causes a key item to no longer be unique.

### Children

- One or more Sectionref definitions. Each sectionref refers to a previously defined section.
- Zero or one HelpLink definitions pointing to the location in a CRF Help document.
- Zero or more KeyItemref references pointing to a section. A KeyItemref definition is only valid in the definition of a repeating form.
- Zero or one Translations definition that specifies a set of translated strings for the translatable attributes of the study component. A Translations element must include one or more Translation definitions. Each Translation element represents the translation of one attribute of a study component into the language of one locale. The Translations element must be the last child element in the study component definition. In a Form definition, you can translate the **MNEMONIC**, **NOTE**, and **TITLE** attributes. If you translate any attributes, include translation definitions for all translatable attributes that are part of the study component definition.

### Example 1

This example illustrates the definition of the Demographics form, which consists of two sections: Demographics (DEM) and Smoking History (SH). The example assumes that individual components and controls are already defined and shows how to define:

- Items that include the components and controls.
- Sections that include the Items.
- A form that includes the sections.

1 Define items for the Demographics section.

```
<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>
```

2 Define items for the Smoking History section.

```
<ITEM REFNAME="SMOKE" QUESTION="Has the subject ever smoked? ">
    <CONTROLREF REFNAME="SMOKERADIO"/>
</ITEM>
<ITEM REFNAME="EVERSMOKED" QUESTION="If the subject has ever smoked, has the
subject quit smoking? ">
    <CONTROLREF REFNAME="SMOKERADIO"/>
</ITEM>
<ITEM REFNAME="WHATSMOKED" QUESTION="If the subject currently smokes, the
subject smokes: ">
    <CONTROLREF REFNAME="SMOKECHECKBOX"/>
</ITEM>
```

3 Define the Demographics and Smoking History sections.

```
<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>
```

4 Define the Demographics form.

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

## Example 2

This example illustrates a form with translations for the TITLE, MNEMONIC, and NOTE attributes.

```
<FORM REFNAME="DEM" TITLE="Demographics" MNEMONIC="DEM"
   NOTE="Use this form to record demographic data and smoking history."
   TYPE="CRF">
   <SECTIONREF REFNAME="DEM"/><SECTIONREF REFNAME="SH"/>
   <TRANSLATIONS>
      <TRANSLATION NAME="TITLE" LOCALE="da-DK" DISPLAYTEXT="Demografi"/>
      <TRANSLATION NAME="TITLE" LOCALE="sy-SE" DISPLAYTEXT="Demografi"/>
      <TRANSLATION NAME="MNEMONIC" LOCALE="da-DK" DISPLAYTEXT="DEM"/>
      <TRANSLATION NAME="MNEMONIC" LOCALE="sy-SE" DISPLAYTEXT="DEM"/>
      <TRANSLATION NAME="NOTE" LOCALE="da-DK"
```

```
        DISPLAYTEXT="Verwenden Sie dieses Formular zur Erfassung von
        demografischen
        Daten und Rauchen Geschichte"/>
     <TRANSLATION NAME="NOTE" LOCALE="sy-SE"
        DISPLAYTEXT="Använd detta formulär för att registrera demografiska data
        och rökning historia"/>
   </TRANSLATIONS>
</FORM>
```

# FormSet

## Purpose

Allows you to group CRFs into:

- Visits
- Common CRFs
- Screening CRFs
- Enrollment CRFs
- Monitoring CRFs
- Status reports
- Associated CRFs

## 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/>
      <TRANSLATIONS/>
</FORMSET>
```

## Attributes

**REFNAME**="*name*"

Name used when referring to the formset in the definition of a study version. This name must be unique among form sets. The formset RefName Conflict is required for all studies. 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.

**TYPE**="VISIT|ENROLLMENT|SCREENING|COMMONCRF|MONITOR|STATUS|REGDOCS|VISITREPORTS|RELATION"

Type of formset. The options are:

- **VISIT**—Set of subject visits.
- **ENROLLMENT**—Set of enrollment CRFs.
- **SCREENING**—Set of screening CRFs.
- **COMMONCRF**—Set of CRFs 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 definition in both a **COMMONCRF** formset and in each **VISIT** formset in which the form should appear.

  Required.

**Warning:** After you define a common CRF by including it in a **COMMONCRF** formset, and data for any subject has been entered in it, you cannot revert the form to regular CRF status by removing it from the **COMMONCRF** formset in the StudyVersion 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 subject. If you attempt to change a study version in either of these ways, you will lose subject 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 definition with a different **REFNAME** from the common CRF, and add it to the appropriate **VISIT** formsets in the study version.

- **MONITOR**—Set of monitoring forms: Regulatory Document checklists and Visit Reports.
- **STATUS**—Not supported.
- **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.
- **REGDOCS**—Not supported.
- **VISITREPORTS**—Not supported.

**UUID**="*id*"

Universally Unique Identifier; a string that identifies the component uniquely across all studies, study 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:** The InForm application and the MedML Installer utility convert alphabetic characters in UUIDs to uppercase.

| Component | UUID |
|---|---|
| Enrollment | d882ce3a-0f42-11d2-a419- 00a0c963e0ac |

| Component | UUID |
|---|---|
| Screening | d882ce38-0f42-11d2-a419- 00a0c963e0ac |
| Common | 9d6bbc5d-5811-11d2-8065- 00a0c9af7674 |
| Conflict | PF_UUID_CONFLICT_FORMSET |

**TITLE**="*text*"

Title of the formset. You can use the **TRANSLATIONS** attribute to provide translations of the attribute value. Required.

**MNEMONIC**="*name*"

Abbreviation of the visit title; can be the same as the **REFNAME** for formsets that define subject visits, this abbreviation appears in the timeline at the top of the visit window. You can use the **TRANSLATIONS** attribute to provide translations of the attribute value. Required.

**LANGUAGE**="*name*"

Code for the culture and language used to display text values. If you do not include a **LANGUAGE** attribute, the default is the installed default product locale. Required if text strings are translated (if a **TRANSLATIONS** element is included as a child of the study component definition).

**Note:** Relying on the default product locale is not recommended, because the default product locale can be changed.

**STARTHOUR**="*n*"

Number of hours from the start of the study (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. If you do not specify a value, the MedML Installer utility orders formsets in the order in which you enter them in the StudyVersion definition. Optional.

**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 study protocol. The options are true or false. True is the default. Optional.

**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 study protocol. The options are true or false. False is the default. Optional.

**DYNAMIC**="*true*|*false*"

Indicates whether the visit is assigned dynamically to a subject based on the value of a data item in a form. The options are 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 The options are true or false. False is the default. Optional.

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 definition consisting of a single item containing a DateTimeControl 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 UUIDs.

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

| Component | UUID |
|---|---|
| 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

- One or more Formref definitions. Each Formref element refers to a previously defined form.
- Zero or one Translations definition that specifies a set of translated strings for the translatable attributes of the study component. A Translations element must include one or more Translation definitions. Each Translation element represents the translation of one attribute of a study component into the language of one locale. The Translations element must be the last child element in the study component definition. In a formset definition, you can translate the MNEMONIC and **TITLE** attributes. If you translate any attributes, include translation definitions for all translatable attributes that are part of the study component definition.

### Example 1

This example illustrates a visit formset for Week 2 of a study. The formset is composed of three forms:

- Vital Signs/Blood Pressure (VS/BP)
- Adverse Experience (AE)
- Compliance (CMP)

For the purpose of the example, the components, controls, items, and sections 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 study version.**

1 Define the Vital Signs/Blood Pressure CRF, which is composed of the Vital Signs (VS) and Blood Pressure (BP) sections.

```
<FORM REFNAME="VSBP" TITLE="Vital Signs/Blood Pressure" MNEMONIC="VS/BP"
    TYPE="CRF">
    <SECTIONREF REFNAME="VS"/>
    <SECTIONREF REFNAME="BP"/>
</FORM>
```

2 Define the Adverse Experience CRF, which has a single repeating section called Adverse Experiences.

```
<FORM REFNAME="AE" TITLE="Adverse Experience" MNEMONIC="AE" TYPE="CRF">
    <SECTIONREF REFNAME="AE"/>
</FORM>
```

3 Define the Compliance CRF, which is composed of the Subject Compliance Information (PCI) and Drug Kit (DK) sections.

```
<FORM REFNAME="CMP" TITLE="Subject Compliance" MNEMONIC="CMP"
    TYPE="CRF">
    <SECTIONREF REFNAME="PCI"/>
    <SECTIONREF REFNAME="DK"/>
</FORM>
```

4 Within a study version definition, define a formset for the Week 2 visit, including the three form

definitions by using Formref elements.

```
<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 study version. In this example, the AE CRF is a common CRF 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 CRFs 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>
```

## Example 3

This example illustrates a formset definition in which the TITLE and MNEMONIC attributes are translated.

```
<FORMSET REFNAME="Visit5" TITLE="Unsched" MNEMONIC="UNSC"
TYPE="Visit" LANGUAGE="en-US" SCHEDULED="false" REPEATING="true">
   <FORMREF REFNAME="DEM" ORDER="1"/>
   <FORMREF REFNAME="VS" ORDER="2"/>
   <FORMREF REFNAME="AE" />
   <TRANSLATIONS>
      <TRANSLATION NAME="TITLE" LOCALE="nl-NL" DISPLAYTEXT="Ongepland"/>
      <TRANSLATION NAME="MNEMONIC" LOCALE="nl-NL" DISPLAYTEXT="ONGEPL"/>
   </TRANSLATIONS>
</FORMSET>
```

# GroupControl

## Purpose

Allows 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
- CheckBoxControl
- GroupControl
- PullDownControl
- RadioControl
- SimpleControl
- TextControl

## 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"]
   [ISCOLLAPSIBLE="true|false">
      <CONTROLREF+ attributes/>
      <TRANSLATIONS/>
</GROUPCONTROL>
```

### Attributes

**REFNAME**="*name*"

Name used when referring to the group control in the definition of another control. This name must be unique among group controls. Required.

**NAME**="*name*"

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

**UUID**="*id*"

Universally Unique Identifier; a string that identifies the component uniquely across all studies, study 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 within the group control: Left, Center, Right, Top, Middle, or Bottom. Left is the default. Optional.

**LAYOUT**="VERTICAL|HORIZONTAL|NOWRAP"

Orientation of the checkboxes: Vertical, Horizontal, or Nowrap. 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. Nowrap is the default. Optional.

**CAPTION**="*text*"

Text that appears on the screen with the group. You can use the **TRANSLATIONS** attribute to provide translations of the attribute value. Optional.

**LANGUAGE**="*name*"

Code for the culture and language used to display text values. If you do not include a **LANGUAGE** attribute, the default is the installed default product locale. Required if text strings are translated (if a **TRANSLATIONS** element is included as a child of the study component definition).

**Note:** Relying on the default product locale is not recommended, because the default product locale can be changed.

**CAPTIONALIGN**="LEFT|RIGHT|TOP|BOTTOM"

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

**ISCOLLAPSIBLE**="*true*|*false*"

Indicates whether the control is a dynamic control. A dynamic control is a control whose visibility is dependent on the selection of its parent control.

- **True**—Control is hidden until the parent control in the group control is selected.
- **False**—Control is always visible.

False is the default. Optional.

## Children

- One or more Controlref definitions. Each Controlref refers to a previously defined component that identifies one item in the group.

  **Note: When defining complex controls with Controlref 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 software 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.**

- Zero or one Translations definition that specifies a set of translated strings for the translatable attributes of the study component. A Translations element must include one or more Translation definitions. Each Translation element represents the translation of one attribute of a study component into the language of one locale. The Translations element must be the last child element in the study component definition. In a GroupControl definition, you can translate the **CAPTION** attribute.

## Example

The group control in this example combines two text controls to capture the two components of a blood pressure measurement. Additionally, the example illustrates the translation of the **CAPTION** attribute of the group control.

1 Define one text control for the first text box and the slash separating the text boxes, and define a second text control for the second text box and the units.

```
<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"/>
```

2 Define a group control that uses Controlrefs to include the two text controls.

```
<GROUPCONTROL REFNAME="BPGRP" NAME="BPGRP"
   LANGUAGE="en-US" CAPTION="Diastolic/Systolic" LAYOUT="HORIZONTAL">
     <CONTROLREF REFNAME="BP1" ORDER="1/>
```

```
      <CONTROLREF REFNAME="BP2" ORDER="2/>
      <TRANSLATIONS>
         <TRANSLATION NAME="CAPTION" LOCALE="es-ES"
         DISPLAYTEXT="Diastólica/Sistólica"/>
      </TRANSLATIONS>
</GROUPCONTROL>
```

# GroupType

## Purpose

Loads the group types supported in the InForm application: ItemGroup, Query, Reporting, and Signature. These GroupTypes are preloaded with your InForm system installation using XML that is internal to Oracle. Do not make any changes to this XML.

# HelpLink

## Purpose

Defines a link from an item on a form to a location in a CRF Help document. To set up the link:

- In the HelpLink definition, point to the location in the CRF Help document that the user will see when clicking underlined item text on a CRF.
- Include the HelpLink definition as a child of the Item definition.

## Syntax

<HELPLINK
DOCREFNAME="*name*"
BODYREFNAME="*name*"
[BOOKMARK="*name*"]/>

## Attributes

**DOCREFNAME**="*name*"

RefName of the Documentation component that defines the CRF Help document. Required.

**BODYREFNAME**="*name*"

RefName of the DocBody component that defines the HTML file containing the information to which the HelpLink points. Required.

**BOOKMARK**="*name*"

Name of an HTML bookmark in the HTM file referenced in the **BODYREFNAME** attribute. Required if you want the Document window to open to the specific item help text when a user clicks an item's underlined text on a CRF; otherwise optional.

## Example

The following example illustrates how to set up a link from a CRF to a specific location in a CRF Help document:

1 Create an HTML file containing the help text you want to display. Identify the start of each segment of text to which the CRF will link with an HTML bookmark. The following example

shows a bookmark from an HTM file in the correct format:

```
<h3><a name="Item2">Item 2  Date of Birth</a></h3>
<ul>
<li>Date of birth must be entered. </li>
<li>Use pull-down menu to supply month, day, and year of birth. </li>
</ul>
```

2 Define a HelpLink that points to the Documentation component for the CRF Help, the DocBody component for the HTML file, and the bookmark within that file. For example:

```
<HELPLINK DOCREFNAME="Study"
BODYREFNAME="DEMHELP"
BOOKMARK="Item2"/>
```

3 In the XML file that defines the CRF, include the HelpLink as a child of the appropriate item:

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

# HTMLTemplate

## Purpose

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 reserved for use by Oracle.

### 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 the 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 studies, study databases, and machines. Required.

**DESCRIPTION**="*text*"

Description of the HTML template. Optional.

**UUID**="*id*"

Universally Unique Identifier; a string that identifies the component uniquely across all studies, study databases, and machines. Optional.

### Example

The following example shows the generated XML code used to load a piece of HTML code used in rendering enrollment CRFs.

<HTMLTEMPLATE
TEMPLATENAME="PF_HTML_ENROLLOVERFRAMESET"
BROWSERNAME="NS0000"
RESOURCEUUID="PF_RESOURCE_NS0000_ENROLLOVERFRAMESET"/>

## Item

### Purpose

Combines previously defined controls into a data point on a CRF. 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"]
   [SDVCRITICAL="true|false"]
   [DISPLAYOVERRIDE="READONLY|EDITABLE|HIDDEN"]>
      <CONTROLREF attributes/>
      <HELPLINK attributes/>
      <TRANSLATIONS/>
</ITEM>
```

### Attributes

**REFNAME**="*name*"

Name used when referring to the Item in the definition of another control. This name must be unique among items. 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.

**QUESTION**="*text*"

Text of the question or prompt that instructs the user to enter a specific data item. You can use the **TRANSLATIONS** attribute to provide translations of the attribute value. Required.

**Note:** You can use HTML elements 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. Use of this attribute is strongly recommended in itemsets. You can use the **TRANSLATIONS** attribute to provide translations of the attribute value. The text specified in the **QUESTION** attribute is the default. Optional.

**UUID**="*id*"

Universally Unique Identifier; a string that identifies the component uniquely across all studies, study databases, and machines. Optional.

**Note:** When you create a repeating formset, most often to identify an unscheduled visit, the first form in the formset must contain an item with a DateTimeControl definition. The item definition must include the following required UUID:

BD991BBF-B0A4-11D2-80E3- 00A0C9AF7674

**LANGUAGE**="*name*"

Code for the culture and language used to display text values. If you do not include a **LANGUAGE** attribute, the default is the installed default product locale. Required if text strings are translated (if a **TRANSLATIONS** element is included as a child of the study component definition).

**Note:** Relying on the default product locale is not recommended, because the default product locale can be changed.

**CALCULATED**="*true*|*false*"

Indicates whether the item contains a CalculatedControl. The options are true or false. False is the default. If you specify that the Calculated attribute is true, the item is unnumbered, to distinguish it from items that require user entry. Optional.

**ITEMREQUIRED**="*true*|*false*"

Indicates whether the item is required for data entry on the form to be complete. The options are true or false. 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. Optional.

**SDVREQUIRED**="*true*|*false*"

Indicates whether the item requires source verification. The options are true or false. 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. Optional.

**SDVCRITICAL**="*true*|*false*"

Indicates whether the item is marked as critical. The options are true or false. False is the default.

If you set the SDVCRITICAL attribute to True for an item that exists in an itemset and outside of an itemset, only the instance of the item that exists outside of the itemset is critical.

**DISPLAYOVERRIDE**="READONLY|EDITABLE|HIDDEN"

Overrides the default item display mode, which is determined by the rights in a user rights group. The options 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 on the form.

**Note:** If the item is in an item group, the display setting for that item group takes precedence over the DISPLAYOVERRIDE setting.

### Children

- One Controlref definition. Each Controlref refers to a previously defined component that identifies a data entry option.
- Zero or one HelpLink 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.
- Zero or one Translations definition that specifies a set of translated strings for the translatable

attributes of the study component. A Translations element must include one or more Translation definitions. Each Translation element represents the translation of one attribute of a study component into the language of one locale. The Translations element must be the last child element in the study component definition. In an item definition, you can translate the **LABEL** and QUESTION attributes. If you translate any attributes, include translation If you translate any If you translate any attributes, include translation definitions for all translatable attributes that are part of the study If you translate any attributes, include translation definitions for all translatable attributes that are part of the study component definition.

## Example 1

This example illustrates the sequence for creating an item that requests information about a subject's smoking habits. Additionally, the example illustrates the translation of the **QUESTION** and **LABEL** attributes.

1 Create PFElement definitions for each checkbox. Note that commonly used components such as the Not Done component can be defined elsewhere and used by reference.

```
<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"/>
```

2 Enclose each PFElement definition in a simple control by using an Elementref element.

```
<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>
```

3 Use the SimpleControl definitions in a checkbox control by using Controlref elements.

```
<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>
```

4 Use the CheckBoxControl definition in the definition of the item by using a Controlref element.

```
<ITEM REFNAME="WHATSMOKED" QUESTION="If the subject currently smokes, the
subject smokes: ">
    <CONTROLREF REFNAME="SMOKECHECKBOX"/>
</ITEM>
```

5 Add a HelpLink definition to point to the CRF Help for the item.

```
<ITEM REFNAME="WHATSMOKED" QUESTION="If the subject currently smokes, the
subject smokes:"
LABEL="Smoking materials" LANGUAGE="en-US">
     <CONTROLREF REFNAME="SMOKECHECKBOX"/>
     <HELPLINK DOCREFNAME="Study" BODYREFNAME="DEMHELP" BOOKMARK="Item9"/>
     <TRANSLATIONS>
        <TRANSLATION NAME="QUESTION" LOCALE="es-ES"
        DISPLAYTEXT="Si el paciente fuma actualmente, el paciente fuma"/>
        <TRANSLATION NAME="LABEL" LOCALE="es-ES" DISPLAYTEXT="Materiales de
        Fumar"/>
        <TRANSLATION NAME="QUESTION" LOCALE="fr-FR" DISPLAYTEXT="Si le patient
```

```
        fume actuellement, le patient fume"/>
        <TRANSLATION NAME="LABEL" LOCALE="fr-FR" DISPLAYTEXT="Fumeur"/>
     </TRANSLATIONS>
</ITEM>
```

### Example 2

This example illustrates the Date of Visit section definition that is required in the first form of a repeating formset. This section definition uses an item definition with a DateTimeControl. The Section, Item, and DateTimeControl definitions all include specific required UUIDs.

```
<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

Combines previously defined items into a group for the purpose of setting their display options within a rights group.

### Syntax

```
<ITEMGROUP
  GROUPNAME="name"
   [GROUPDESCRIPTION="name"]
   <ITEMREF attributes/>
</ITEMGROUP>
```

### Attributes

**GROUPNAME**="*name*"

Name used when referring to the item group in the definition of an ItemGroupRef. This name must be unique among item groups. Required.

**GROUPDESCRIPTION**="*name*"

Description of the item group. Optional.

### Children

An ItemGroup description can include any number of Itemref descriptions. Each Itemref element identifies one item to be included in the item group.

### 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

Includes a previously defined item group within the definition of a rights group. An ItemGroupref appears only as the child of a rights group in which it is included; it is not submitted as a stand-alone component.

### Syntax

```
<ITEMGROUPREF
```

REFNAME="*name*"
DISPLAYOVERRIDE="READONLY|EDITABLE|HIDDEN"/>

### Attributes

**REFNAME**="*name*"

Name used when referring to the item group in the definition of a RightsGroup. This name must be unique among item groups. Required.

**DISPLAYOVERRIDE**="READONLY|EDITABLE|HIDDEN"

Overrides the item-level display mode of the items in the item group. The item-level display mode is set with the DISPLAYOVERRIDE attribute of the Item 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.

Required.

### Example

The following example illustrates the inclusion of the CRC_Hidden item group in the CRC RG rights group.

```
<ITEMGROUP
 GROUPNAME="CRC_Hidden"
  GROUPDESCRIPTION="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

- Includes previously defined items in the definition of a section, itemset, or item group. Include one Itemref element for each item in the section, itemset, or item group definition.
- Specifies the RefName of an item as the location of a mapped control in the Path element of a mapping definition. Include one Itemref element in a RefName path defined by a Path element.

An itemref appears only as the child of a section or path 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 specified item. Required.

**KEYITEM**="*true*|*false*"

Indicates whether the specified itemis a key item in an itemset definition. Only items with text, numeric, or date/time values can be key items. Items with compound controls, including checkbox controls, group controls, radio controls, or drop-down lists cannot be key items. Options are true or false. False is the default. Optional.

**Note:** You can specify key items for repeating forms by using the KeyItemref element. 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 visit, form, and itemset. 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. Options are true or false. False is the default. Optional.

**ORDER**="*n*"

Sequence in which each itemref appears in the Section definition. 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. Optional.

**Note:** When including an Itemref definition in a Path element, only the **REFNAME** attribute is valid.

## Examples

This example illustrates the use of itemrefs in the definition of a section called Duration of Hypertension, which contains two items. For more information, see *Section* (on page 177).

```
<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 element. 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 elements.

```
<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

Defines a set of items to be used in a form with repeating information; for example, an Adverse Event or Concomitant Medication form. Each set of items appears in a single row of the form.

- You can only add an ItemSet component to a Section definition with the REPEATING attribute set to true.
- Only one ItemSet definition can appear in a REPEATING Section definition.
- Regular, non-repeating Item definitions cannot appear in a REPEATING Section definition.

## Syntax

```
<ITEMSET
   REFNAME="name"
    [DESIGNNOTE="text"]
    [UUID="id"]
    [ITEMREQUIRED="true|false"]
    [SDVREQUIRED="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. This name must be unique among ItemSet definitions. 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; a string that identifies the component uniquely across all studies, study databases, and machines. Optional.

**ITEMREQUIRED**="*true*|*false*"

Indicates whether the itemset is required for data-entry on the CRF to be complete. The options are true or false. True is the default. If you specify false, the itemset is gray on the CRF after submission to indicate that entry is not required. Optional.

**SDVREQUIRED**="*true*|*false*"

Indicates whether the itemset requires source verification. The options are true or false. True is the default. If you specify false, the itemset is gray on the Source Verification view of the CRF to indicate that source verification is not required. Optional.

**SDVCRITICAL**="*true*|*false*"

Indicates whether the itemset is marked as critical. The options are true or false. False is the default.

**DISPLAYOVERRIDE**="READONLY|EDITABLE|HIDDEN"

Overrides the default display itemset mode, which is determined by the rights in a user rights group. Values are:

- **READONLY**—Users can view but not enter data in the itemset, regardless of their rights groups.
- **EDITABLE**— Users can edit data in the itemset, regardless of their rights groups.
- **HIDDEN**—Users cannot see the itemset, regardless of their rights groups. On the CRF, itemset rows are renumbered and the itemset does not appear 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. The options are true or false. False is the default. Optional.

- For a key item within the itemset that has UNIQUEKEY=true, then that one item must be unique across all itemset rows.
- When checking itemset key item uniqueness, deleted itemset rows are considered.
- Items within itemsets cannot be repeating form key items.

If you specify true:

- If two rows of an itemset are submitted in the same visit and form with any key items having the same value, the InForm software rejects the input of the second instance.
- The combination of all of the key items must be unique. An error occurs if all key items in one itemset row are the same as the key items in another itemset row.
- The setting does not override the UNIQUEKEY attribute of the key items; they still need to be individually unique. As a result, it does not make sense to specify true for both the ITEMSET and ITEMREF elements.

**REVISION**="*number*"

Revision number for the itemset.

**INITIALROWCOUNT**="*number*"

Number of rows in the itemset.

**ROWCOUNTFIXED**="*true*|*false*"

Indicates whether the itemset contains fixed rows.

### Children

An ItemSet definition must include one or more Itemref definitions. Each itemref refers to a previously defined item.

### 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 in the itemset. Use the **QUESTION** attribute of the Item definition to specify the text of the question as it appears on the CRF where users enter data; use the **LABEL** attribute to specify the text of the column heading in the tabular CRF where data is displayed.

```
<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>
```

2 Use Itemref definitions to group the Item definitions in an ItemSet definition.

```
<ITEMSET REFNAME="AE">
    <ITEMREF REFNAME="AE_SYMPTOM"/>
    <ITEMREF REFNAME="AE_START"/>
    <ITEMREF REFNAME="AE_END"/>
</ITEMSET>
```

3 Create a Section definition that includes the itemset by using an Itemref definition. The **REPEATING** attribute of the Section definition must be set to true.

```
<SECTION REFNAME="AESECTION"
    TITLE="Adverse Events"
    REPEATING="true">
        <ITEMREF REFNAME="AE"/>
</SECTION>
```

# ItemSetCell

## Purpose

Indicate whether a specific cell in a Repeating Data itemset has a predefined value, is blank, or predefined and blank.

## Syntax

```
<ITEMSETCELL
   ITEMREFNAME="name"
   ITEMSETROW="number"
   PREDEFINEDVALUEREFNAME="name"
   ISBLANK="true|false"
</ITEMSETCELL>
```

## Attributes

**ITEMREFNAME**="*name*"

RefName of the item associated with the blank cell. Required.

**ITEMSETROW**="*number*"

Row number in the itemset in which the cell exists. Required.

**PREDEFINEDVALUEREFNAME="***name*"

RefName of the control in the cell that contains a predefined value in the itemset. Optional.

**ISBLANK**="*true*|*false*"

Indicates whether the cell in the itemset is blank. Optional.

# KeyItemref

## Purpose

Identifies an item in a repeating form as a key item for the purpose of navigating through instances of the form. The values of key items are displayed in a drop-down list in the InForm application to allow users to navigate to a specific instance of the form. A KeyItemref appears only as the child of a form 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.
- Drop-down lists.
- Radio groups.

  Items defined as radio groups cannot be key items if they contain any type of nested compound controls, such as checkbox controls, group controls, or other radio groups.
- Text boxes.

**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 element that includes an Item element in an ItemSet definition. Items in an itemset cannot 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 in which the item that the KeyItemref references exists. Required.

**ITEMREFNAME**="*name*"

RefName of the item that the KeyItemref includes in the definition of repeating form. Required.

**ORDER**="*n*"

Sequence in which each key item appears in the navigational drop-down list on the repeating form. 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 definition. Optional.

**UNIQUEKEY**="*true*|*false*"

Indicates whether the key item must be unique within the visit and form. The options are true or false. False is the default. Optional.

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.

If you set a single item KEYITEMREF to have UNIQUEKEY=true, that item must be unique across all form instances.

### 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>
```

# MedMLData

## Purpose

The first and last required element in an XML file that defines study definition components. It wraps all other elements, signaling to the MedML Installer utility that the enclosed elements consist of study 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 MedML Installer utility. The required version text is "PhaseForward-MedML- Inform4".

## Children

The child components of the MedMLData element define components of a study. They include components for defining visits, forms, rules and events, resources, and study administration data.

## Example

The following example illustrates the use of the MedMLData element as a wrapper for the MedML elements that define the DOV form.

```
<MEDMLDATA xmlns="PhaseForward-MedML-Inform4">

  <DATETIMECONTROL REFNAME="DOV" CAPTIONALIGN="LEFT"ALIGN="LEFT"
    STARTYEAR="2002" ENDYEAR="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>
```

# PFElement

## Purpose

Defines a component such as an item in a drop-down list or an option in a list of radio buttons. PFElement is the lowest-level CRF component you can define. To use a PFElement element, you must enclose it in the definition of another control such as a drop-down list or a simple control.

**Note:** Unlike other CRF components, PFElement 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"]>
      <TRANSLATIONS/>
</PFELEMENT>
```

### Attributes

**REFNAME**="*name*"

Name used when referring to the PFElement in the definition of another control. This name must be unique among PFElements. A RefName can have a maximum of 63 characters. 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.

**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. You can use the **TRANSLATIONS** attribute to provide translations of the attribute value. Required.

**LANGUAGE**="*name*"

Code for the culture and language used to display text values. If you do not include a **LANGUAGE** attribute, the default is the installed default product locale. Required if text strings are translated (if a **TRANSLATIONS** element is included as a child of the study component definition).

**Note:** Relying on the default product locale is not recommended, because the default product locale can be changed.

**TYPE**="STRING|INTEGER|FLOAT"

Data type of the element. String, Integer, or Float. String is the default. Required.

**Note:** All the PFElements used in a single multiple selection control, such as a checkbox control, 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; a string that identifies the component uniquely across all studies, study databases, and machines. Optional.

### Children

Zero or one Translations definition that specifies a set of translated strings for the translatable attributes of the study component. A Translations element must include one or more Translation definitions. Each Translation element represents the translation of one attribute of a study component into the language of one locale. In a PFElement definition, you can translate the **LABEL** attribute.

### Example

The following set of elements defines two PFElements that can later be used as components in a list. Additionally, the example illustrates the translation of the **LABEL** attribute.

```
<PFELEMENT REFNAME="Y" LABEL="Yes" LANGUAGE="en-US"
   TYPE="STRING" VALUE="Y">
      <TRANSLATIONS>
         <TRANSLATION NAME="LABEL" LOCALE="fr-FR" DISPLAYTEXT="Oui"/>
      </TRANSLATIONS>
</PFELEMENT>
<PFELEMENT REFNAME="N" LABEL="No" LANGUAGE="en-US"
   TYPE="STRING"VALUE="N">
      <TRANSLATIONS>
         <TRANSLATION NAME="LABEL" LOCALE="fr-FR" DISPLAYTEXT="Non"/>
      </TRANSLATIONS>
</PFELEMENT>
```

## PulldownControl

### Purpose

Defines a drop-down list in which a user can select a single item. Drop-down lists are composed of previously defined PFElements that you include by using Elementrefs. Drop-down lists can also contain previously defined units that you include by using Unitrefs.

### 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/>
      <UNITREF attributes/>
      <TRANSLATIONS/>
</PULLDOWNCONTROL>
```

### Attributes

**REFNAME**="*name*"

Name used when referring to the drop-down list in the definition of another control. This name must be unique among PullDownControls. 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.

**NAME**="*name*"

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

**UUID**="*id*"

Universally Unique Identifier; a string that identifies the component uniquely across all studies, study databases, and machines. Optional.

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

Alignment of the PFElements within the control. The options are Left, Center, Right, Top, Middle, or Bottom. Left is the default. Optional.

**CAPTION**="*text*"

Text that appears on the screen with the drop-down list. You can use the **TRANSLATIONS** attribute to provide translations of the attribute value. Optional.

**LANGUAGE**="*name*"

Code for the culture and language used to display text values. If you do not include a **LANGUAGE** attribute, the default is the installed default product locale. Required if text strings are translated (if a **TRANSLATIONS** element is included as a child of the study component definition).

**Note:** Relying on the default product locale is not recommended, because the default product locale can be changed.

**CAPTIONALIGN**="LEFT|RIGHT|TOP|BOTTOM"

Position of the caption relative to the calculated value. The options are Left, Right, Top, or Bottom. Left is the default. Optional.

**UNITDISPLAYTYPE**="ELEMENT"

Type of control used to display units included in the PullDownControl with a Unitref definition. Element is the only valid value.

### Children

- One or more Elementref definitions. Each Elementref refers to a previously defined PFElement that identifies one item in the drop-down list.
- Zero or one Unitref definition, which refers to a previously defined unit.

> **Note:** When defining compound controls with Elementref 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 software 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.

- Zero or one Translations definition that specifies a set of translated strings for the translatable attributes of the study component. A Translations element must include one or more Translation definitions. Each Translation element represents the translation of one attribute of a study component into the language of one locale. The Translations element must be the last child element in the study component definition. In a PulldownControl definition, you can translate the **CAPTION** attribute.

### Example

The PullDownControl in this example consists of a list of reasons for prescribing medication. Additionally, the example illustrates the translation of the **CAPTION** attribute.

To create the list:

1 Define a PFElement for each medication reason.

```
<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"/>
```

2 Define a PullDownControl that includes each PFElement definition by using an Elementref.

```
<PULLDOWNCONTROL REFNAME="MEDREASON"
   NAME="MEDREASON" LANGUAGE="en-US" CAPTION="Reason">
      <ELEMENTREF REFNAME="ADVERSE" ORDER="1"/>
      <ELEMENTREF REFNAME="CHRONIC" ORDER="2"/>
      <ELEMENTREF REFNAME="HORMONE" ORDER="3"/>
      <ELEMENTREF REFNAME="PROPHYL" ORDER="4"/>
      <TRANSLATIONS>
         <TRANSLATION NAME="CAPTION" LOCALE="sy-SE" DISPLAYTEXT="Anledningen"/>
         <TRANSLATION NAME="CAPTION" LOCALE="nn-NO" DISPLAYTEXT="Grunnen"/>
      </TRANSLATIONS>
</PULLDOWNCONTROL>
```

# QueryGroup

## Purpose

Specifies a set of users who can close queries initiated by other members of the same group.

**Note:** You cannot create query groups in the user interface. You must use MedML to define query groups.

**Note:** The only way to remove a user from a group is through 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; a string that identifies the component uniquely across all studies, study databases, and machines. Optional.

## Children

A QueryGroup definition can include zero or more Userref definitions. Each Userref refers to a previously created User definition that identifies one InForm user.

## Example

In the following QueryGroup, users named Kevin and George can each close queries initiated by the other user.

```
<QUERYGROUP GROUPNAME="CRA Query">
   <USERREF USERNAME="Kevin"/>
   <USERREF USERNAME="George"/>
</QUERYGROUP>
```

# RadioControl

## Purpose

Defines a list of radio buttons from which users must select one option. Radio controls are composed of previously defined components that you include by using Controlrefs. You can include the following types of components in a RadioControl definition:

- CalculatedControl
- CheckBoxControl
- GroupControl
- PullDownControl
- RadioControl
- SimpleControl
- TextControl

## 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/>
      <TRANSLATIONS/>
</RADIOCONTROL>
```

### Attributes

**REFNAME**="*name*"

Name used when referring to the RadioControl in the definition of another control. This name must be unique among radio controls. 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.

**NAME**="*name*"

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

**UUID**="*id*"

Universally Unique Identifier; a string that identifies the component uniquely across all studies, study databases, and machines. Optional.

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

Alignment of the components included with Controlrefs within the radio control. The options are Left, Center, Right, Top, Middle, or Bottom. Left is the default. Optional.

**LAYOUT**="VERTICAL|HORIZONTAL|NOWRAP"

Orientation of the radio buttons. The options are Vertical, Horizontal, or Nowrap. 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. Optional.

**CAPTION**="*text*"

Text that appears on the screen with the radio button list. You can use the **TRANSLATIONS** attribute to provide translations of the attribute value. Optional.

**LANGUAGE**="*name*"

Code for the culture and language used to display text values. If you do not include a **LANGUAGE** attribute, the default is the installed default product locale. Required if text strings are translated (if a **TRANSLATIONS** element is included as a child of the study component definition).

**Note:** Relying on the default product locale is not recommended, because the default product locale can be changed.

**CAPTIONALIGN**="LEFT|RIGHT|TOP|BOTTOM"

Position of the caption relative to the radio control. The options are Left, Right, Top, or Bottom. Left is the default. Optional.

### Children

- One or more Controlref 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 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 software 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.**

- Zero or one Translations definition that specifies a set of translated strings for the translatable attributes of the study component. A Translations element must include one or more Translation definitions. Each Translation element represents the translation of one attribute of a study component into the language of one locale. The Translations element must be the last child element in the study component definition. In a RadioControl definition, you can translate the **CAPTION** attribute.

### Example 1

This example continues the SimpleControl example, which demonstrates how to define a set of PFElements for specifying gender and wrap them in SimpleControls. The example shows how to use Controlrefs to include the previously defined simple controls in a radio control. Additionally, the example illustrates the definition of a translation for the **CAPTION** attribute.

1 Create PFElements that define selections for Male and Female.

```
<PFELEMENT REFNAME="MALE" LABEL="Male" TYPE="STRING" VALUE="MElement"/>
<PFELEMENT REFNAME="FEMALE" LABEL="Female" TYPE="STRING" VALUE="FElement"/>
```

2 Create SimpleControls that use an Elementref to include the definition of each PFElement.

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

3 Using Controlrefs to include each SimpleControl, create a RadioControl.

```
<RADIOCONTROL REFNAME="GENDER"
   NAME="GENDERRADIO"
   LANGUAGE="en-US"
   CAPTION="Select one:"
   LAYOUT="HORIZONTAL">
      <TRANSLATIONS>
         <TRANSLATION NAME="CAPTION" LOCALE="pt-PT" DISPLAYTEXT="Selecione um"/>
      </TRANSLATIONS>
      <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 study subject did not complete the study.

### Components of the list:

- This list includes several items that you can simply define as PFElements and then enclose in simple control definitions:
  - Did not meet criteria for randomization.
  - Adverse experience (other than adverse laboratory experience).
  - Adverse laboratory experience.
  - 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:
  - Subject decision.
  - Physician decision.
  - Sponsor request.

  For these items, define the caption that appears next to a radio button as a PFElement, and enclose it in a simple control. Define the text box and its caption as a text control. Then, create a group control that combines the two components.
- The Death item consists of the caption that appears next to a radio button and a group of three drop-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 simple control. Define the three date components as a datetime control, and enclose the datetime control in a group control. Finally, define a group control that includes the simple control for the caption and the group control for the date.

### Definition steps:

1 Define the PFElements you need, and enclose them in SimpleControl definitions.

```
<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>
```

2 Define text controls for the Other (specify) and Reason items. Note the use of the HTML special character ** ** to control the spacing.

```
<TEXTCONTROL REFNAME="OTHER" NAME="OTHER"
    HEIGHT="1" LENGTH="32" MAXLENGTH="50"
    DATATYPE="STRING"
    CAPTION="Other (specify): "
    CAPTIONALIGN="TOP"/>
<TEXTCONTROL REFNAME="REASONTEXT" NAME="REASONTEXT"
    HEIGHT="1" LENGTH="32" MAXLENGTH="50"
    DATATYPE="STRING"
    CAPTION="      Reason: "
    CAPTIONALIGN="LEFT"/>
```

3 Define group controls for the Subject decision, Physician decision, and Sponsor text boxes and their captions.

```
<GROUPCONTROL REFNAME="CRITERIAGROUP" UUID="PF_SC_CRITIERIA"
    NAME="CRITERIAGROUP" LAYOUT="HORIZONTAL">
    <CONTROLREF REFNAME="CRITERIA" ORDER="1"/>
</GROUPCONTROL>
<GROUPCONTROL REFNAME="AEGROUP" UUID="PF_SC_AE"
    NAME="AEGROUP" LAYOUT="HORIZONTAL">
    <CONTROLREF REFNAME="AE" ORDER="1"/>
</GROUPCONTROL>
<GROUPCONTROL REFNAME="ALEGROUP" UUID="PF_SC_ALE"
    NAME="ALEGROUP" LAYOUT="HORIZONTAL">
    <CONTROLREF REFNAME="ALE" ORDER="1"/>
</GROUPCONTROL>
<GROUPCONTROL REFNAME="PDECGROUP" UUID="PF_SC_PATIENTDECISION"
    NAME="PDECGROUP" LAYOUT="HORIZONTAL"
    ALIGN="LEFT">
    <CONTROLREF REFNAME="PD" ORDER="1"/>
    <CONTROLREF REFNAME="REASONTEXT" ORDER="2"/>
</GROUPCONTROL>
<GROUPCONTROL REFNAME="PHDECGROUP" UUID="PF_SC_PHYSICIANDECISION"
    NAME="PHDECGROUP" LAYOUT="HORIZONTAL"
    ALIGN="LEFT">
    <CONTROLREF REFNAME="PHD" ORDER="1"/>
    <CONTROLREF REFNAME="REASONTEXT" ORDER="2"/>
</GROUPCONTROL>
```

```
<GROUPCONTROL REFNAME="SDECGROUP" UUID="PF_SC_SPONSORDECISION"
    NAME="SDECGROUP" LAYOUT="HORIZONTAL"
    ALIGN="LEFT">
    <CONTROLREF REFNAME="SD" ORDER="1"/>
    <CONTROLREF REFNAME="REASONTEXT" ORDER="2"/>
</GROUPCONTROL>
<GROUPCONTROL REFNAME="OTHERGROUP" UUID="PF_SC_OTHER"
    NAME="OTHERGROUP" LAYOUT="HORIZONTAL">
    <CONTROLREF REFNAME="OTHER" ORDER="1"/>
    <CONTROLREF REFNAME="REASONTEXT" ORDER="2"/>
</GROUPCONTROL>
<GROUPCONTROL REFNAME="LOSTGROUP" UUID="PF_SC_LOST"
    NAME="DEATHGROUP" LAYOUT="HORIZONTAL">
    <CONTROLREF REFNAME="LOST" ORDER="1"/>
</GROUPCONTROL>
```

4 Define a datetime control for the Death item.

```
<DATETIMECONTROL REFNAME="COMMONDATE" NAME="COMMONDATE"
    STARTYEAR="1997"    ENDYEAR="2002"
    DISPLAYDAY="true"     DISPLAYMONTH="true"
    DISPLAYYEAR="true"     REQUIREMONTH="true"
    REQUIREDAY="true"     REQUIREYEAR="true"/>
```

5 Define a group control that includes the DEATHDATE group control and the simple control that defines the Death caption.

```
<GROUPCONTROL REFNAME="DEATHGROUP" UUID="PF_SC_DEATH"
    NAME="DEATHGROUP" LAYOUT="HORIZONTAL">
    <CONTROLREF REFNAME="DEATHSC" ORDER="1"/>
    <CONTROLREF REFNAME="COMMONDATE" ORDER="2"/>
</GROUPCONTROL>
```

6 Define a radio control that includes a Controlref representing each simple or compound control in the list.

```
<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>
```

# ReportingGroup

## Purpose

**Note:** You cannot create reporting groups in the user interface. You must use MedML to define reporting groups.

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 the InForm Ad Hoc Reporting workspace

**Note:** 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 reporting group Required.

**GROUPDESCRIPTION**="*name*"

Text describing the reporting group Optional.

## Children

A ReportingGroup definition can include zero or more Userref definitions. Each Userref refers to a previously created User definition that identifies one InForm 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

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**; a string that identifies the component uniquely across all studies, study 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*"

Code for the culture and language used to display text values. If you do not include a **LANGUAGE** attribute, the default is the installed default product locale. Required if text strings are translated (if a **TRANSLATIONS** element is included as a child of the study component definition).

> **Note:** Relying on the default product locale is not recommended, because the default product locale can be changed.

**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.

### 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"/>
```

## ReviewStage

### Purpose

Creates a custom review stage associated with a custom review state for use in the Data Viewer. Use with the ReviewState element to associate a custom review stage with its parent custom review state. You must create three stages for each review state.

For example, to track data that requires sponsor review, you can create a review state called **Review State 1**, which contains the stages **Needs Review**, **Review Pending**, and **Review Complete**. Navigate to the Data Viewer to view the number of data items in each state.

**Notes:**

**You must use MedML and the MedML Installer utility to define custom review states and custom review stages. You cannot create them in the InForm user interface.**

**You do not need to define all five custom review states at one time, but for each custom review state that you define, you must define all three associated custom review stages at one time.**

### Syntax

```
<REVIEWSTAGE
   REFNAME="name"
   [STAGE="n"]
   [LABEL="name"]
   [MNEMONIC="name"]
   [LANGUAGE="name"]
      <TRANSLATIONS/>
</REVIEWSTAGE>
```

### Attributes

**REFNAME**="*name*"

RefName of the custom review stage. Required.

**STAGE**="*n*"

One through three, indicating the order in which to display the custom review stage within the column for its parent custom review state in the InForm application. You must specify three custom review stages for each custom review state. Required.

**LABEL**="*name*"

The name for the custom review stage, which appears in hover Help and drop-down lists in the InForm application. Required.

**MNEMONIC**="*name*"

Abbreviated name for the custom review stage, which appears in column headings in the Data Viewer. Required.

**LANGUAGE**="*name*"

Code for the culture and language used to display text values. If you do not include a **LANGUAGE** attribute, the default is the installed default product locale. Required if text strings are translated (if a **TRANSLATIONS** element is included as a child of the study component definition). Required.

**Note:** **Relying on the default product locale is not recommended, because the default product locale can be changed.**

### Example

```
<REVIEWSTAGE REFNAME="ReviewStage1" STAGE="0" LABEL="Needs Review"
MNEMONIC="RS1" LANGUAGE="en-US">
   <TRANSLATIONS>
      <TRANSLATION NAME="LABEL" DISPLAYTEXT="要レビュー" LOCALE="ja-JP" />
<TRANSLATION NAME="MNEMONIC" DISPLAYTEXT="レビューS1" LOCALE="ja-JP" />
</TRANSLATIONS>
   </REVIEWSTAGE>
   <REVIEWSTAGE REFNAME="ReviewStage2" STAGE="1" LABEL="Pending" MNEMONIC="RS2"
   LANGUAGE="en-US">
      <TRANSLATIONS>
         <TRANSLATION NAME="LABEL" DISPLAYTEXT="保留" LOCALE="ja-JP" />
<TRANSLATION NAME="MNEMONIC" DISPLAYTEXT="レビューS2" LOCALE="ja-JP" />
</TRANSLATIONS>
```

```
   </REVIEWSTAGE>
   <REVIEWSTAGE REFNAME="ReviewStage3" STAGE="2" LABEL="Reviewed" MNEMONIC="RS3"
   LANGUAGE="en-US">
      <TRANSLATIONS>
         <TRANSLATION NAME="LABEL" DISPLAYTEXT="レビュー済" LOCALE="ja-JP" />
<TRANSLATION NAME="MNEMONIC" DISPLAYTEXT="レビューS3" LOCALE="ja-JP" />
</TRANSLATIONS>
   </REVIEWSTAGE>
```

# ReviewState

## Purpose

Creates a review state for use in the Data Viewer. The review states provide flexibility in tracking the review process for data. You can create as many as five review states, which you can customize as necessary. Each review state includes three stages, which are also customizable.

For example, to track data that requires sponsor review, you can create a review state called **Review State 1**, which contains the stages **Needs Review**, **Review Pending**, and **Review Complete**. Navigate to the Data Viewer to view the number of data items in each state.

**Notes:**

**You must use MedML and the MedML Installer utility to define custom review states and custom review stages. You cannot create them in the InForm user interface.**

**You do not need to define all five custom review states at one time, but for each custom review state that you define, you must define all three associated custom review stages at one time.**

## Syntax

```
<REVIEWSTATE
   REFNAME="name"
   STATE="n"
   ACTIVATED="true|false"
   LABEL="name"
   MNEMONIC="name"
   LANGUAGE="name">
   <TRANSLATIONS/>
</REVIEWSTATE>
```

## Attributes

**REFNAME**="*name*"

RefName of the review state. Required.

**STATE**="*n*"

One through five, indicating the order in which to display the review state in the user interface. Required.

**ACTIVATED**="*true*|*false*"

True or false, indicating whether to make the review state available in the InForm application. Required.

**LABEL**="*name*"

The name for the custom review stage, which appears in hover Help and drop-down lists in the

InForm application. Required.

**MNEMONIC**="*name*"

Abbreviated name for the custom review stage, which appears in column headings in the Data Viewer. Required.

**MNEMONIC**="*name*"

Abbreviated name with which to refer to the review state. Required.

**LANGUAGE**="*name*"

Code for the culture and language used to display text values. If you do not include a **LANGUAGE** attribute, the default is the installed default product locale. Required if text strings are translated (if a **TRANSLATIONS** element is included as a child of the study component definition). Required.

**Note:** Relying on the default product locale is not recommended, because the default product locale can be changed.

## Children

- Three ReviewStage elements, which define the stages of the review state. For example, each review state can include the following review stages:
  - Needs Review
  - Review Pending
  - Review Complete

## Example

```
<REVIEWSTATE REFNAME="DataReview1" STATE="1" ACTIVATED="true" LABEL="Data
Review" MNEMONIC="R1" LANGUAGE="en-US">
   <REVIEWSTAGE REFNAME="ReviewStage1" STAGE="0" LABEL="Needs Review"
   MNEMONIC="RS1" LANGUAGE="en-US">
      <TRANSLATIONS>
         <TRANSLATION NAME="LABEL" DISPLAYTEXT="要レビュー" LOCALE="ja-JP" />
   <TRANSLATION NAME="MNEMONIC" DISPLAYTEXT="レビューS1" LOCALE="ja-JP" />
   </TRANSLATIONS>
      </REVIEWSTAGE>
      <REVIEWSTAGE REFNAME="ReviewStage2" STAGE="1" LABEL="Pending"
      MNEMONIC="RS2" LANGUAGE="en-US">
         <TRANSLATIONS>
            <TRANSLATION NAME="LABEL" DISPLAYTEXT="保留" LOCALE="ja-JP" />
   <TRANSLATION NAME="MNEMONIC" DISPLAYTEXT="レビューS2" LOCALE="ja-JP" />
   </TRANSLATIONS>
      </REVIEWSTAGE>
      <REVIEWSTAGE REFNAME="ReviewStage3" STAGE="2" LABEL="Reviewed"
      MNEMONIC="RS3" LANGUAGE="en-US">
         <TRANSLATIONS>
            <TRANSLATION NAME="LABEL" DISPLAYTEXT="レビュー済" LOCALE="ja-JP" />
   <TRANSLATION NAME="MNEMONIC" DISPLAYTEXT="レビューS3" LOCALE="ja-JP" />
   </TRANSLATIONS>
      </REVIEWSTAGE>
<TRANSLATIONS>
<TRANSLATION NAME="LABEL" DISPLAYTEXT="データレビュー" LOCALE="ja-JP" />
<TRANSLATION NAME="MNEMONIC" DISPLAYTEXT="レビュー1" LOCALE="ja-JP" />
</TRANSLATIONS>
   </REVIEWSTATE>
```

# Rightref

## Purpose

Allows you to include previously defined rights in the definition of a rights group. A Rightref appears only as the child of a rights group 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 definition.

## Example

The following definition of the PF Admin Rights Group, which identifies the activities that a study administrator can perform, illustrates how to use Rightrefs to include Right definitions in a rights group.

```
<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="Kevin"/>
</RIGHTSGROUP>
```

# RightsGroup

## Purpose

Allows you to create a set of rights and to assign the set of rights to users. The InForm software is delivered with a set of predefined rights and rights groups.

## Syntax

```
<RIGHTSGROUP
  GROUPNAME="name"
   [GROUPDESCRIPTION="text"]
   [UUID="id"]>
   <RIGHTREF* attributes/>
   <USERREF* attributes/>
   <ITEMGROUPREF* attributes/>
```

</RIGHTSGROUP/>

### Attributes

**GROUPNAME**="*name*"

Name of the rights group. Required.

**GROUPDESCRIPTION**="*text*"

Text describing the rights group. Optional.

**UUID**="*id*"

Universally Unique Identifier; a string that identifies the component uniquely across all studies, study databases, and machines. Optional.

### Children

A RightsGroup definition can include zero or more:

- Rightref definitions. Each Rightref refers to a previously created right definition that identifies an InForm activity that users can perform.
- Userref definitions. Each Userref refers to a previously created user definition that identifies one InForm user.
- ItemGroupref definitions. Each ItemGroupref refers to a previously created item group definition that identifies a group of items 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 study 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 Sponser User"/>
   <RIGHTREF RIGHT="DeActivate Sponser 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 and the CRC_Hidden ItemGroupref.

```
<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" />
  <ITEMGROUPREF REFNAME="CRC_Hidden" DISPLAYOVERRIDE="HIDDEN"/>
</RIGHTSGROUP>
```

# Right

### Purpose

Loads definitions of rights that identify specific InForm activities that users can perform. The InForm software 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"/>
```

# Rule

## Purpose

Defines a rule and the script that executes when it fires, and to associate the rule with an event.

## 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 in the AttachRuleSet element. Can be up to 63 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; a string that identifies the component uniquely across all studies, study 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. The options are true or false. True is the default. Optional.

**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 software substitutes values defined as rule arguments for each item or itemset into the help text. This allows 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. The options are:

- **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 checkboxes 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 to assign in a calculated control.
- **Clintrialderivation**—The rule is used in a Clintrial study 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 study.

Required.

**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. The options are true or false. Required if the rule is a Clintrial study 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. Must be 240 characters or less. Do not specify if the rule is not a Clintrial rule. Optional.

**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 definitions are allowed. Rulearg definitions contain parameters that are passed into a rule script.
- One Event definition is required to define the action that results if a rule fails. To associate an event with a rule, use the Eventref element.

### Example

The following rule definition includes the evtGeneric event definition with an Eventref element. Additionally, Rulearg elements are used to define the arguments min, max, qtext, and addpath.

```
<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>
```

# Rulearg

## Purpose

Passes 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 element 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 insert into the rule. Required.

**TYPE**="STRING|NUMERIC|FLOAT|DATE"

Data type for the rule. The options are String, Numeric, Float or Date. String is the default. Required.

## Example

The following example associates the Height Range Checking rule with 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 software 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, the 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

Defines a grouping, under a section heading, of related items or one itemset on a form.

## Syntax

```
<SECTION
   REFNAME="name"
   [DESIGNNOTE="text"]
   [TITLE="text"]
   [UUID="id"]
   [NOTE="text"]
   [LANGUAGE="name"]
   [REPEATING="true|false"]>
      <ITEMREF+ attributes/>
      <TRANSLATIONS/>
</SECTION>
```

## Attributes

**REFNAME**="*name*"

Name used when referring to the section in the definition of a form. This name must be unique among sections and must not exceed 31 characters. 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.

**TITLE**="*text*"

Text of the section title, displayed at the top of the section on the screen. You can use the **TRANSLATIONS** attribute to provide translations of the attribute value. Optional.

**UUID**="*id*"

Universally Unique Identifier; a string that identifies the component uniquely across all studies, study databases, and machines. Optional.

**Note:** When you create a repeating ***FormSet*** (on page 124), 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. You can use the **TRANSLATIONS** attribute to provide translations of the attribute value. Optional.

**LANGUAGE**="*name*"

Code for the culture and language used to display text values. If you do not include a **LANGUAGE** attribute, the default is the installed default product locale. Required if text strings are translated (if a **TRANSLATIONS** element is included as a child of the study component definition).

**Note:** Relying on the default product locale is not recommended, because the default product locale can be changed.

**REPEATING**="*true*|*false*"

Indicates whether the section contains an itemset definition. The options are true or false. False is the default. Optional.

Note the following rules about 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 definitions cannot appear in a REPEATING section definition.

### Children

- One or more Itemref definitions. Each Itemref refers to a previously defined item or itemset.

**Note:** When you create a repeating ***FormSet*** (on page 124), often to identify an unscheduled visit, the first form in the formset must contain a Date of Visit section definition consisting of one item—a DateTimeControl definition that includes the month, day, year, hour, and minute of the visit. The Section, Item, and DateTimeControl definitions must include specific UUIDs, as noted in the Example section.

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

- Zero or one Translations definition that specifies a set of translated strings for the translatable attributes of the study component. A Translations element must include one or more Translation definitions. Each Translation element represents the translation of one attribute of a study component into the language of one locale. The Translations element must be the last child element in the study component definition. In a Section definition, you can translate the **NOTE** and **TITLE** attributes. If you translate any attributes, include translation definitions for all translatable attributes that are part of the study component definition.

## Example 1

This example illustrates the creation of a section called Duration of Hypertension. The section is composed of two Itemrefs:

- The first Itemref refers to an item that consists of a question and a radio control. For the purpose of this example, assume that a Yes/No radio control has been defined previously.
- The second Itemref refers to an item that consists of a question and a group control. The group control is composed of two text controls, each of which includes a Unit definition by using a Unitref element.

1 Create Unit definitions for years and months.

```
<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"/>
```

2 Enclose the years and months Units in text control definitions by using Unitref elements.

```
<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>
```

3 Create a GroupControl definition by using the text controls.

```
<GROUPCONTROL REFNAME="HTDURATIONGRP" NAME="HTDURATIONGROUP"
    LAYOUT="HORIZONTAL">
    <CONTROLREF REFNAME="HTYEARSTXT" ORDER="1/>
    <CONTROLREF REFNAME="HTMONTHSTXT" ORDER="2/>
</GROUPCONTROL>
```

4 Create Item definitions for the two line items in the section.

```
<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>
```

5 Use Itemrefs to enclose the Item definitions in the definition of the Duration of Hypertension

section.

```
<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 124). The section must contain only one item—the DOV item, which allows users to enter the date and time of the visit. We recommend that you create this section as a separate form. Additionally, the example illustrates the translation of the **TITLE** and **NOTE** attributes.

```
<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"
  NOTE="Complete this form before any others in the visit."
  LANGUAGE="en-US"
  UUID="BD991BBE-B0A4-11D2-80E3-00A0C9AF7674">
    <ITEMREF REFNAME="DOV" ORDER="1"/>
    <TRANSLATIONS>
      <TRANSLATION NAME="TITLE" LOCALE="fr-FR" DISPLAYTEXT="Date de la
      visite"/>
      <TRANSLATION NAME="NOTE" LOCALE="fr-FR"
        DISPLAYTEXT="Remplissez ce formulaire avant tout autre à la
        visite."/>
    </TRANSLATIONS>
</SECTION>
<FORM REFNAME="DOV" TITLE="Date of Visit" MNEMONIC="DOV" TYPE="CRF">
  <SECTIONREF REFNAME="DOV"/>
</FORM>
```

## Sectionref

### Purpose

- Includes previously defined sections in the definition of a form.
- Includes one Sectionref element for each section in the form.
- Specifies the RefName of a section as the location of a mapped control in the Path element of a mapping definition.
- Includes one Sectionref element in a RefName path defined by a Path element.

A sectionref appears only as the child of a Form or Path element 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 that the Sectionref is including in the definition of a form or path. Required.

**ORDER**="*n*"

Sequence in which each sectionref appears in the form definition. 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. Optional.

**Note:** When including a Sectionref definition in a Path element, 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: Demographics (DEM) and Smoking History (SH).

```
<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>
```

# Sequence

## Purpose

Specifies a numbering sequence. Sequences allow sponsors to indicate how numbers are assigned to screened subjects, enrolled subjects, queries, and other types of numbered data. Each sequence is associated with a sequence type, defined with the SequenceType element.

## Syntax

```
<SEQUENCE
  UUID="id"
   SEQUENCENAME="name"
   SEQUENCETYPENAME="name"/>
```

## Attributes

**UUID**="*id*"

Universally Unique Identifier; a string that identifies the component uniquely across all studies, study databases, and machines. Required.

**Note:** The InForm 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 sequence type to which the sequence number belongs. Required.

## Example

The following example shows the XML elements 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

Defines the types of entities for which the InForm application generates sequential numbers. When you create a definition of the numerical sequence by using the Sequence element, you specify a type to which the sequence belongs by including a SequenceType definition. The InForm software defines the following types of sequence numbers:

- Screening
- Enrollment
- Query
- Randomization

## Syntax

```
<SEQUENCETYPE
  [UUID="id"]
   SEQUENCETYPENAME="name"/>
```

## Attributes

**UUID**="*id*"

Universally Unique Identifier; a string that identifies the component uniquely across all studies, study 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 software:

```
<SEQUENCETYPE SEQUENCETYPENAME="Enrollment"/>
<SEQUENCETYPE SEQUENCETYPENAME="Screening"/>
<SEQUENCETYPE SEQUENCETYPENAME="Query"/>
<SEQUENCETYPE SEQUENCETYPENAME="Randomization"/>
```

# SignatureGroup

## Purpose

Specifies a set of users who can sign case books or CRFs.

**Notes:**

**You cannot create signature groups in the user interface. You must use MedML to define signature groups.**

**The only way to remove a user from a group is in 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/>
      <TRANSLATIONS/>
</SIGNATUREGROUP>
```

## Attributes

**GROUPNAME**="*name*"

Name of the signature group. Required.

**GROUPDESCRIPTION**="*text*"

Text describing the signature group. Optional.

**UUID**="*id*"

Universally Unique Identifier; a string that identifies the component uniquely across all studies, study databases, and machines. Optional.

**LANGUAGE**="*name*"

Code for the culture and language used to display text values. If you do not include a **LANGUAGE** attribute, the default is the installed default product locale. Required if text strings are translated (if a **TRANSLATIONS** element is included as a child of the study component definition).

**Note: Relying on the default product locale is not recommended, because the default product locale can be changed.**

**CRFTEXT**="*text*"

Text of the Electronic Signature Affidavit displayed to members of the signature group when signing a CRF associated with the signature group. 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 CRF.
- 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. You can use the **TRANSLATIONS** attribute to provide translations of the attribute value. Optional.

**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. 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. Optional.

**CRFMEANING**="*text*"

Text that summarizes the meaning of the signature on the CRF. This text is displayed on the Signature Details page and in the list of completed and required signatures on the CRF. You can use the **TRANSLATIONS** attribute to provide translations of the attribute value.

**Note:** The text for the meaning of a signature is stored in the language of the site where the CRF was signed. The language does not change if a subject is transferred to another site with a different site study locale.

**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. 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. You can use the **TRANSLATIONS** attribute to provide translations of the attribute value. Optional.

**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. 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. Optional.

**CRBMEANING**="*text*"

Text that summarizes the meaning of the signature on the case book. This text is displayed on the Signature Details page and in the list of completed and required signatures on the CRF used for signing a case book. You can use the **TRANSLATIONS** attribute to provide translations of the attribute value.

**UPDATE**="*true*|*false*"

Perform an incremental update for the specified signature group by modifying only the properties that you specify.

## Children

- Zero or more Userref definitions. Each Userref refers to a previously created User definition that identifies one InForm user.
- Zero or one Translations definition that specifies a set of translated strings for the translatable attributes of the study component. A Translations element must include one or more Translation definitions. Each Translation element represents the translation of one attribute of a study component into the language of one locale. The Translations element must be the last child element in the study component definition. In a SignatureGroup definition, you can translate the **CRFTEXT**, **CRBTEXT**, **CRFMEANING**, and **CRBMEANING** attributes. If you translate any attributes, include translation definitions for all translatable attributes that are part of the study component definition.

> **Note:** Translations are not supported for the **CRFFILE** and **CRBFILE** attributes.

### Example

In the following SignatureGroup, users dobrien and lhill can sign.

```
<SIGNATUREGROUP GROUPNAME="CRA Signature">
  <USERREF USERNAME="dobrien"/>
  <USERREF USERNAME="lhill"/>
</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. Additionally, the SignatureGroup definition includes the definition of a translation string for the CRFTEXT attribute.

```
<SIGNATUREGROUP GROUPNAME="PI Signature" LANGUAGE="en-US"
   CRFTEXT="&lt;font name='Arial' size='2'&gt;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.&lt;br&gt;&lt;br&gt;
   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.&lt;br&gt;&lt;br&gt;
   To this I do attest by supplying my user name and password and clicking
   the button marked &t b &gt;Submit&lt;/b&gt; below.&lt;/font&gt;"
   CRFMEANING="Approval">
      <USERREF USERNAME="dobrien"/>
      <USERREF USERNAME="lhill"/>
      <TRANSLATIONS>
         <TRANSLATION NAME="CRFTEXT" LOCALE="fr-FR"
         DISPLAYTEXT="&lt;font name='Arial' size='2'&gt;Par ma signature datée
         ci-dessous,
         Moi, %s %s, je confirme que ce cas formulaire de rapport affiche avec
         précision
         les résultats des examens, des tests, des évaluations et des
         constatées dans les traitements. &lt;br&gt; &lt;br&gt;
         Conformément à la Section de 11,100 Titre 21 du Code of Federal
         Règlement, il s'agit de certifier que j'ai l'intention que ce
         signature électronique doit être juridiquement contraignant équivalent
         de ma signature manuscrite. &lt;br&gt; &lt;br&gt;
         Pour cela, je ne témoignent par la fourniture de mon nom d'utilisateur
         et mot de passe et cliquez sur
         le bouton &lt; b &gt;Submit&lt;/b&gt; ci-dessous.&lt;/font&gt;
         <TRANSLATION NAME="CRFMEANING" LOCALE="fr-FR"
         DISPLAYTEXT="Approbation"/>
      </TRANSLATIONS>
</SIGNATUREGROUP>
```

# SignCRF

## Purpose

Identifies forms that require signature and specifies which signature group a user must belong to in order to be able to sign the CRF.

## Syntax

```
<SIGNCRF
  SIGNATUREGROUPNAME="name"
  FORMREFNAME="name"
  [RESETFORMSTATE="true|false"]
  [INVALIDATIONLEVEL="USER|GROUP"]
  [FINALCRF="true|false"]/>
```

## Attributes

**SIGNATUREGROUPNAME**="*name*"

RefName of a signature group that contains users who are authorized to sign the form specified in the FORMREFNAME attribute. Required.

**FORMREFNAME**="*name*"

RefName of a CRF 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 CRFs 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 CRF that is fully signed resets the state of the CRF to not fully signed. When a CRF is reset, the original signatures remain valid. However, the CRF must now be signed by a member of the newly-associated signature group as well. False is the default. Optional.

**INVALIDATIONLEVEL**="USER|GROUP"

Specifies whether a signature should be invalidated when a data item is imported after the CRF has been signed, for example, when a coded value is imported with the Central Coding application. The options are:

- **USER**—The CRF or case book signature is invalidated if the user who signed can view the item being imported.
- **GROUP**—The CRF 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. Optional.

**FINALCRF**="*true*|*false*"

Indicates whether signing the form specified in the **FORMREFNAME** attribute signs entire the

case book. False is the default. Optional.

### Example

The following example illustrates the use of the SignCRF element to designate the DEM, VS, and SC forms as requiring signature by a member of the CRA Signature group. The FINALCRF attribute indicates that signing the SC form signs the case report 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

Defines an option in a compound control such as a list of radio buttons or of checkboxes. Using simple controls allows 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 simple controls:

- CheckBoxControl
- GroupControl
- Item
- RadioControl

### 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/>
      <UNITREF attributes/>
      <TRANSLATIONS/>
</SIMPLECONTROL>
```

### Attributes

**REFNAME**="*name*"

Name used when referring to the simple controls in the definition of another control. This name must be unique among simple controls. 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.

**NAME**="*name*"

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

**UUID**="*id*"

Universally Unique Identifier; a string that identifies the component uniquely across all studies, study databases, and machines. Optional.

**CAPTION**="*text*"

Text that appears on the screen with the simple control. This caption provides the opportunity to define additional text to accompany the caption defined for a PFElement. You can use the **TRANSLATIONS** attribute to provide translations of the attribute value. Optional.

**LANGUAGE**="*name*"

Code for the culture and language used to display text values. If you do not include a **LANGUAGE** attribute, the default is the installed default product locale. Required if text strings are translated (if a **TRANSLATIONS** element is included as a child of the study component definition).

**Note:** Relying on the default product locale is not recommended, because the default product locale can be changed.

**CAPTIONALIGN**="LEFT|RIGHT|TOP|BOTTOM"

Position of the caption relative to the calculated value. The options are Left, Right, Top, or Bottom. Left is the default. Optional.

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

Alignment of the PFElement within the control. The options are Left, Center, Right, Top, Middle, or Bottom. Left is the default. Optional.

**UNITDISPLAYTYPE**="ELEMENT"

Type of control used to display units included in the simple control with a Unitref definition: Element is the only valid value.

### Children

- One Elementref definition, which refers to a previously defined PFElement.
- Zero or one Unitref definition, which refers to a previously defined unit.
- Zero or one Translations definition that specifies a set of translated strings for the translatable attributes of the study component. A Translations element must include one or more Translation definitions. Each Translation element represents the translation of one attribute of a study component into the language of one locale. The Translations element must be the last child element in the study component definition. In a SimpleControl definition, you can translate the **CAPTION** attribute.

### Example

This example demonstrates how to define a set of PFElements for specifying gender and wrap them in simple controls. The simple controls are then available for inclusion in a compound control such

as a radio control.

1 Create PFElements that define selections for "Male" and "Female."

```
<PFELEMENT REFNAME="MALE" LABEL="Male" TYPE="STRING" VALUE="MElement"/>
<PFELEMENT REFNAME="FEMALE" LABEL="Female" TYPE="STRING" VALUE="FElement"/>
```

2 Create SimpleControls that use an Elementref to include the definition of each PFElement.

```
<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 *RadioControl* (on page 158).

# Site

## Purpose

Defines a study location.

**Note:** You cannot define sites in the user interface. You must use MedML to define site groups.

## 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"]>
   [SVAUTOSELECTRATE="num"]
   [SVFIRSTNSUBJECTS="num"]
   [SITESERVER="server name"]
   [SITEDATEFORMAT="MONTH_DAY_YEAR|DAY_MONTH_YEAR|YEAR_MONTH_DAY"]
   [STUDYLOCALE="text"]
   [USERNAMEORDER="F,L|L,F"]>
</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.

**STARTDATE**="*date*"

Date that the site came online. 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" |

Required.

**ENDDATE**="*date*"

Date that the site came offline. For example, the date that the last subject 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 subject 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.

**STUDYLOCALE**="*text*"

Code for the preferred study locale of the site. The study locale is the locale in which the study metadata is defined, including visit names, CRF names, section labels, questions, and control labels. The value must match the study locale for a study version in the study. Required.

**Note:** After you add a subject to a site, you cannot change the site study locale from the user interface or through MedML.

**SVAUTOSELECTRATE**="n"

Number that corresponds to the percent of forms marked SV Required, which you must source verify.

**SVFIRSTNSUBJECTS**="n"

The number of subjects to source verify in order, based on the time when the subject is assigned the status Enrolled. For example, to indicate that you must source verify the first seven subjects enrolled in a study, specify SVFIRSTNSUBJECTS=7.

**USERNAMEORDER**="F,L|L,F"

Order in which the user's given and surnames are presented in a signature affidavit. The default is the user name order specified for the study on the System Configuration page. The order specified for the site overrides the order specified for the study.

- **F,L**—Given name followed by surname.
- **L,F**—Surname followed by given name.

Optional.

**UPDATE**="*true*|*false*"

Perform an incremental update for the specified site by modifying only the properties that you specify.

## Example

The following example defines a site for Clínica Ortopédica in Valencia, Spain, including a user whose Username is Dr Cortina.

```
<SITE NAME="Clínica Ortopédica"
   MNEMONIC="CORTO"
   ADDRESS="2150 Avenida Universidad"
   ADDRESS2="Oficina 14B"
   CITY="Valencia"
   STATE="ES"
   PHONE="(+34) 96-555-55-55"
   FAX="(+34) 96-555-55-99
   EMAIL="drcortina@clinicaortopedica.com"
   TIMEZONE="CET"
   STARTDATE="10/23/2008"
   STUDYLOCALE="es-ES"
   USERNAMEORDER="L,F">
</SITE>
```

# SiteGroup

## Purpose

Specifies a group of users who have access to a named site.

**Note:** 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. This name corresponds to the **NAME** attribute in the Site definition for the site. Required.

## Children

A SiteGroup definition can include zero or more Userref definitions. Each Userref refers to a previously created User definition that identifies one InForm 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

Defines a study sponsor. The Sponsor definition is for documentation purposes.

**Note:** You must use MedML to define sponsors. You cannot perform this task from the user interface.

## 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.

**Note:** The name attribute is required if you plan to generate PDF files using the CRF Submit software.

**PROGRAM**="*text*"

Name of the study. Optional.

**THERAPEUTICAREA**="*text*"

Therapeutic area of the study. Optional.

**NOTE**="*text*"

Description of the study. Optional.

**ADDRESS**="*addr1*"

First line of the sponsor address. Optional.

**ADDRESS2**="*addr2*"

Second line of the sponsor address. Optional.

**CITY**="*name*"

City of the sponsor address. Optional.

**STATE**="*name*"

State of the sponsor address. Optional.

**PROVINCE**="*name*"

Province of the sponsor address. Optional.

**ZIPCODE**="*code*"

Sponsor zip code. Optional.

**POSTCODE**="*code*"

Sponsor postal code. Optional.

**COUNTRY**="*name*"

Country of the sponsor address. Optional.

**PHONE**="*num*"

Sponsor telephone number. Optional.

**ALTPHONE**="*num*"

Sponsor alternate telephone number. Optional.

**FAX**="*num*"

Sponsor fax number. Optional.

**EMAIL**="*addr*"

Sponsor email address. Optional.

**CONTACTUSERREF**="*username*"

Name of the user who is the primary sponsor contact. Optional.

**LANGUAGE**="*name*"

Language used in correspondence with the sponsor. English is the default. Optional.

**LOGOFILE**="*file*"

Filename of the sponsor logo file. Optional.

**LOGOTYPE**="GIF|JPEG|TEXT"

File type of the sponsor logo file. The options are 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

Assigns a document, as defined in a Documentation element, to a study version.

## Syntax

<STUDYVERSIONDOC
VERSIONDESCRIPTION="*n*"
DOCREFNAME="*name*"
[ORDER="*n*"]/>

## Attributes

**VERSIONDESCRIPTION**="*n*"

Number of the study version with which to associate the document. Required.

**DOCREFNAME**="*name*"

RefName of the document, as specified in the Documentation element. Required.

**[ORDER**="*n*" **]**

Order in which the tab representing the document will appear in the Document or Help window. The default is the order in which the document is referenced in the XML file with the StudyVersionDocs definition. Optional.

## Example

The following excerpt from a MedMLData definition file illustrates the use of the StudyVersionDoc element to set the StudyVersion 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"/>
```

# StudyVersionSite

## Purpose

Records the study version that a site is currently using. This component needs to be updated each time a study version changes in a way that does not require Institutional Review Board (IRB) approval, and whenever IRB approval is received for a study version change that does require approval.

**Note:** Before you can use the StudyVersionSite component to specify the study version that a site is using, you must run the XML file that loads site definitions.

## Syntax

<STUDYVERSIONSITE
VERSIONDESCRIPTION="*n*"
[SITENAME="*name*"]
[SITEMNEMONIC="*name*"]
[ACCEPTDATE="*date*"]/>

## Attributes

**VERSIONDESCRIPTION**="*n*"

Number of the study version currently in use at the site. This number must match the **VERSIONDESCRIPTION** attribute with which the study version is defined. Required.

**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 is defined.

**SITEMNEMONIC**="*name*"

Abbreviated name of the site. This name must match the **MNEMONIC** attribute with which the site is defined. Either the **SITENAME** or the **SITEMNEMONIC** attribute is required.

**ACCEPTDATE**="*date*"

Date of IRB approval of a study version change.

**Note:** The **ACCEPTDATE** attribute is for documentation purposes and to differentiate study versions for repeating form display. The actual date that a site moves to a new study version 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

Groups formsets together under a single version of a study and its protocol. Each time you need to implement a revised form or study document, you create a new StudyVersion definition, update the StudyVersionSite definition for each site where the changed form version applies, and update the StudyVersionDoc definition for each applicable study 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; a string that identifies the component uniquely across all studies, study databases, and machines. Optional.

**STUDYNAME**="*name*"

Name of the study. Required.

**VERSIONDESCRIPTION**="*text*"

Description of the study version. For example, an indication of what was changed in this version. Required.

**PROTOCOL**="*name*"

Name and version number of the IRB-approved study protocol. Optional.

**SPONSORDATE**="*date*"

Date of sponsor approval of the study version, 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 name for the drug being studied. Optional.

## Children

A StudyVersion definition must include one or more FormSet definitions. Each formset defines a visit or other group of forms and specifies which forms to include in the set. Because formsets are not reusable, they are defined inside the definition of a study version.

**Warning:** If the StudyVersion definition includes one or more common CRFs, consider that after you define a common CRF by including it in a COMMONCRF formset, and data for any subject has been entered in it, you cannot revert the form to regular CRF status by removing it from the COMMONCRF formset in the StudyVersion 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 subject. If you attempt to change a study version in either of these ways, you will lose subject 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 definition with a different RefName from the common CRF, and add it to the appropriate VISIT formsets in the study version.

## 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>
```

# SVCriticalForm

### Purpose

Adds a form to the Critical Forms list.

### Syntax

```
<SVCRITICALFORM
    FORMREFNAME="name"
    SITEMNEMONIC="name"
    CRITICAL="true|false"
</SVCRITICALFORM>
```

### Attributes

**FORMREFNAME**="*name*"

RefName of the form to add to the Critical Forms list. Required.

**SITEMNEMONIC**="*name*"

Mnemonic for the site with which the form is associated. Required.

**CRITICAL**="*true*|*false*"

Indicates whether the form is marked as critical for source verification. Options are true or false. Required.

# SVCriticalItem

### Purpose

Overrides the SVCRITICAL setting for the ITEM.

### Syntax

```
<SVCRITICALITEM
ITEMREFNAME="name"
CRITICAL="study default|true|false"
FORMREFNAME="name"
SITEMNEMONIC="name"
</SVCRITICALITEM>
```

### Attributes

**ITEMREFNAME**="*name*"

RefName of the item to mark as critical. Required.

**CRITICAL**="s*tudy default*|*true*|*false*"

Indicates whether to mark the item as critical for source verification. The options are study default, true, or false. Required.

The study default option allows you to reset a previously-defined override for the item. If you specify study default for the CRITICAL attribute, the SDVCRITICAL attribute of the ITEM element determines the critical setting for the item.

**FORMREFNAME**="*name*"

RefName of the form on which the item to mark as critical exists. Optional.

**SITEMNEMONIC**="*name*"

Mnemonic of the site with which the item is associated. Optional.

- To mark an item as critical across all sites, use the SVCRITICALITEM element, and do not specify a SITEMNEMONIC.
- To mark an item as critical for one site, do one of the following:
  - Use the SVCRITICAL element, and specify a SITEMNEMONIC.
  - Use the SVCRITICAL setting for the ITEM.
  - Use the InForm user interface.

## SysConfig

### Purpose

Specifies the setting of an InForm configuration variable.

### Syntax

```
<SYSCONFIG
    CONFIGNAME="name"
    TYPE="0"
    VALUE="text" />
```

### Attributes

**CONFIGNAME**="*name*"

Name of the configuration variable. Required.

The InForm application has the following configuration variables:

- **AllowPasswordReuse**—1 (yes) or 0 (no), indicating whether users can change to a previously used password when performing password updates. The default is 1.
- **AutoAnswerManualQueries**—1 (on) or 0 (off), indicating whether the InForm application 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.
- **DaysPasswordExpiration**—Number of days that can pass before the InForm software requires users to change their passwords. The default is 30.
- **DISPLAYTRAFFICLIGHTS**—1 (yes) or 0 (no), indicating whether to use traffic light icons as status indicators. Read only.
- **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. Read only.
- **EnrollWithIncompleteForms**—1 (yes) or 0 (no), indicating whether the InForm application permits a subject 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.
- **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.
- **LDAPBDN**—An entry in the LDAP hierarchy that is used for InForm and reporting authentication.
- **LDAPSERVER**—The name of the server on which the Oracle Directory Server is installed.
- **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 software requires a user to log in again. The default is 5.
- **MinutesReIdentification**—Number of minutes that a session can be active before the InForm software requires a user to log in again. The default is 120.
- **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 subject numbers. Read only.
- **PostQueryForConflictResolution**—1 (yes) or 0 (no), indicating whether to create a query when during synchronization data is found to be entered into a data item by two different servers. The default is 1.
- **QueryMaxLength**—Maximum number of characters of query text displayed below an item on a CRF. The default is 350.
- **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).
- **RandomizationSrc**—Name of the randomization source manager (COM object) that accesses the default randomization source database. The default name is "Inform.PFRandomization.1." Read only.
- **RandSimpleCentral**—Sequence number format for Simple Central randomization schemes (one central drug kit list from which numbers are assigned sequentially). Read only.
- **RandSimpleCentralSRC**—Name of the randomization source manager (COM object) that accesses the randomization source database for Simple Central randomization schemes. Read only.
- **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). Read only.
- **RandSimpleSiteSRC**—Name of the randomization source manager (COM object) that accesses the Simple Site randomization source database. Read only.
- **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). Read only.
- **RandStratifiedBySiteSRC**—Name of the randomization source manager (COM object) that accesses the Site Stratification randomization source database. Read only.
- **REPORTINGINTERNALURI**—Cognos parameter that is set when running the CRNConfig installer. It is an internal URI that the InForm server uses to communicate to the Cognos server. The information can be found in cogstartup.xml. Example: http://appsru02.north.pf.com:9300/p2pd/servlet/dispatch.
- **REPORTINGSERVER**—The URL for the Cognos 8 BI Web service.
- **REPORTINGAUTHENTICATIONNAMESPACE**—The LDAP namespace that is used to authenticate InForm users on the reporting server.
- **REPORTINGUSERROOT**—The top-level reporting folder for the company. Use this field only if you are hosting several companies on one reporting server and have set up reporting

folders for each company. Leave this field blank if you are not hosting studies for different companies, or if you have not set up separate reporting folders for each company.

- **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. Read only.
- **SponsorEditFrozen**—1 (yes) or 0 (no), indicating whether sponsors can edit a form after it has been marked as frozen. The default is 0.
- **SSLFlag**—1 (on) or 0 (off), indicating whether Secure Socket Layer is enabled to provide encryption of data. The default is 1.

  **Note:** Before this option can take effect, you must stop and restart the study.

- **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 server directory used to download data listings with the Listings button. Read only.
- **UniqueIntlDOBSwtch**—Study, site or none, indicating whether the InForm application requires a unique combination of subject initials and date of birth for a study, 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 study.

  **Note:** If you specify that unique initials and date of birth is not required and subjects 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. If you plan to allow the transfer of subjects from one site to another, be aware that if a user transfers a subject to a site where another subject exists with the same initials and date of birth, and the study does not require unique initials and date of birth or only requires site uniqueness, the subject transfer fails. The user must change the subject initials to make the combination unique.

  To prevent this situation, set the UniqueIntlDOBSwitch attribute to require unique initials and DOB across the study.

- **UniquePatIDSwtch**—Study, site, or none, indicating whether the InForm application requires a unique subject ID for a study, a site or not at all:
  - 0—Subject ID is not required to be unique.
  - 1—Subject ID must be unique within a site.
  - 2 (default)—Subject ID must be unique within a study.

  **Notes:**

  If you are using the Data Viewer in the InForm application, you must configure your system to require unique subject IDs within each site or each study.

If you specify that unique initials and date of birth is not required and subjects 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. If you plan to allow the transfer of subjects from one site to another, be aware that if a user transfers a subject to a site where another subject exists with the same initials and date of birth, and the study does not require unique initials and date of birth or only requires site uniqueness, the subject transfer fails. The user must change the subject initials to make the combination unique.

To prevent this situation, set the UniqueIntIDOBSwitch attribute to require unique initials and DOB across the study.

- **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. Read only.
- **Virtual_DownloadDirectory**—Full path name of the InForm server virtual directory used to download data listings with the Listings button. Read only.

**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 the 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

```
<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="NumOfExePlanListenThreads" TYPE="0" VALUE="4"/>
<SYSCONFIG CONFIGNAME="MaxNumOfResubmissions" TYPE="0" VALUE="2"/>
<SYSCONFIG CONFIGNAME="RequireCommentForNA" 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="2"/>
<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="QUERYSELECTION" VALUE="QUERYACTION_CREATEOPEN"/>
```

# Translation

## Purpose

Specifies a translated text string that is associated with an attribute of a study component, along with the locale in which the translated text is used. Use one Translation element for each combination of attribute and locale that is needed in the study version. If you translate any attributes, include translation definitions for all translatable attributes that are part of the study component definition.

The Translation element is a child of the Translations element.

## Syntax

```
<TRANSLATION>
   NAME="attributename"
   LOCALE="text"
   DISPLAYTEXT="text"
   />
</TRANSLATION>
```

## Attributes

**NAME**="*attributename*"

Name of the study component attribute for which translated text is being supplied. Required.

**LOCAL**E="*text*"

Code for the language and culture of the translated text. Required.

**DISPLAYTEXT="text"**

Translated text string. Required.

## Examples

The following example illustrates the translation of an item question and label into Spanish and German.

```
<ITEM refname="itmCardio"
   LANGUAGE="en-US"
   QUESTION="Cardiovascular"
   LABEL="Cardio"
   CALCULATED="false"
   ITEMREQUIRED="true"
   SDVREQUIRED="true"
      <CONTROLREF REFNAME="PEFstatus"/>
      <TRANSLATIONS>
         <TRANSLATION NAME="QUESTION" LOCALE="es-ES" DISPLAYTEXT="Sistema
         cardiovascular"/>
         <TRANSLATION NAME="QUESTION" LOCALE="de-DE" DISPLAYTEXT="Herz-
         Kreislauf-System"/>
         <TRANSLATION NAME="LABEL" LOCALE="es-ES" DISPLAYTEXT="Cardiovascular"/>
```

```
        <TRANSLATION NAME="LABEL" LOCALE="de-DE" DISPLAYTEXT="Herz-Kreislauf"/>
     </TRANSLATIONS>
<ITEM/>
```

# Translations

## Purpose

Defines a set of translated strings for the translatable attributes of a study component. A Translations definition is a child of the study component for which it provides translated strings. The Translations element must be the last child element in the study component definition. If you translate any attributes, include translation definitions for all translatable attributes that are part of the study component definition.

**Note: When you use a Translations element, the LANGUAGE attribute of the associated study component is required.**

## Syntax

```
<TRANSLATIONS>
   <TRANSLATION* attributes/>
</TRANSLATIONS>
```

## Children

A Translations element must include one or more Translation definitions. Each Translation element represents the translation of one attribute of a study component into the language of one locale.

## Examples

The following example illustrates the translation of an item question and label into Spanish and German.

```
<ITEM refname="itmCardio"
   LANGUAGE="en-US"
   QUESTION="Cardiovascular"
   LABEL="Cardio"
   CALCULATED="false"
   ITEMREQUIRED="true"
   SDVREQUIRED="true"
      <CONTROLREF REFNAME="PEFstatus"/>
      <TRANSLATIONS>
         <TRANSLATION NAME="QUESTION" LOCALE="es-ES" DISPLAYTEXT="Sistema
         cardiovascular"/>
         <TRANSLATION NAME="QUESTION" LOCALE="de-DE" DISPLAYTEXT="Herz-
         Kreislauf-System"/>
         <TRANSLATION NAME="LABEL" LOCALE="es-ES" DISPLAYTEXT="Cardiovascular"/>
         <TRANSLATION NAME="LABEL" LOCALE="de-DE" DISPLAYTEXT="Herz-Kreislauf"/>
      </TRANSLATIONS>
<ITEM/>
```

# Unit

## Purpose

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"]>
      <TRANSLATIONS/>
</UNIT>
```

## Attributes

**REFNAME**="*name*"

Name used when referring to the unit in the definition of another control. This name must be unique among units. Required.

**Note:** A text control defined with **TYPE="STRING"** cannot have a unit. A text control 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; a string that identifies the component uniquely across all studies, study databases, and machines. Optional.

The following unit definitions have predefined UUIDs that are not required.

| 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 |

| Unit | UUID |
|---|---|
| Kilogram | 498100f8-e9df-11d1-9e60- 00a0c9769a33 |
| Pound | 498100fc-e9df-11d1-9e60- 00a0c9769a33 |

**SYMBOL**="*text*"

Label used to represent the unit on the screen. You can use the **TRANSLATIONS** attribute to provide translations of the attribute value. Required.

**CLASSIFICATION="text"**

Type of measurement represented by the unit. The following classifications have been predefined:

- Length
- Weight
- Temperature
- Pressure
- Volume
- Area
- Time
- Frequency

Required.

**LANGUAGE**="*name*"

Code for the culture and language used to display text values. If you do not include a **LANGUAGE** attribute, the default is the installed default product locale. Required if text strings are translated (if a **TRANSLATIONS** element is included as a child of the study component definition).

**Note:** Relying on the default product locale is not recommended, because the default product locale can be changed.

**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.

## Children

Zero or one Translations definition that specifies a set of translated strings for the translatable attributes of the study component. A Translations element must include one or more Translation definitions. Each Translation element represents the translation of one attribute of a study

component into the language of one locale. In a Unit definition, you can translate the **SYMBOL** attribute.

### 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

Includes the definition of a unit 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
- PullDownControl
- SimpleControl
- TextControl

## Syntax

<UNITREF
  REFNAME="*name*"/>

## Attributes

**REFNAME**="*name*"

RefName of the unit that the Unitref is including in the control definition. Required.

**Note:** When defining compound controls with ControlRef 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 software 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 definitions of pounds and kilograms are included in a TextControl definition by using Unitref elements.

1 Create Unit definitions for pounds and kilograms. In this example, kilograms are the base units for conversion.

```
<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"/>
```

2 Create a TextControl definition that uses Unitref elements to refer to the Unit definitions for pounds and kilograms. In this example, the unit selections are displayed in a drop-down list.

```
<TEXTCONTROL REFNAME="txtVSWeight"
    LANGUAGE="English" NAME="txtVSWeight" ALIGN="LEFT" CAPTIONALIGN="LEFT"
    UNITDISPLAYTYPE="RADIO" HEIGHT="1" LENGTH="6" MAXLENGTH="5"
DATATYPE="FLOAT"
    PRECISION="1" MINVALUE="0" MAXVALUE="0" MINPROPERTY="NOTSPECIFIED"
    MAXPROPERTY="NOTSPECIFIED">
      <UNITREF REFNAME="POUNDS" />
      <UNITREF REFNAME="KILOGRAMS" />
```

```
</TEXTCONTROL>
```

# Update_Form_Section

## Purpose

Updates a section within a form without modifying the study version. This XML is used to modify an existing section on a form. It cannot be used to add a new section. If subject 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.

**Note:** To determine the RefNames for the CRF and section to update, Oracle recommends that you run the InForm RefName Report in the Central Designer software. For more information, see the Central Designer *InForm Designer's Guide*.

When deleting items with this element, 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 is preferable to remove them by updating the study version for the study.

## 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 to modify. Required.

**FORM_REVISION**="*number*"

The revision number of the form to modify. Required.

**SECTION_REFNAME**="*name*"

The RefName of the section to modify. Required.

**SECTION_REVISION**="*number*"

The revision number of the section to modify. Required.

## Updating a section definition using the Update_Form_Section element

1 Stop the study and make sure that no one can access the study or modify the study 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 MedML Installer utility validates that the form RefName and revision is correct, and that the form contains the section you updated.

7 Restart the study.

8 Test the updated form version after applying this change. The MedML Installer utility does not validate the item metadata against any existing subject data.

## Update_Section_Item

### Purpose

Updates an item within a section without modifying the study version. This XML is used to modify an existing item. It cannot be used to add a new item. If subject 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.

**Note:** To determine the RefNames for the section and item to update, Oracle recommends that you run the InForm RefName Report in the Central Designer software. For more information, see the Central Designer *InForm Designer's Guide*.

**WARNING:** Making changes to study definitions without creating a new study version violates Good Clinical Practice (GCP) and can cause problems with study 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 itemset. Required.

### Updating a section definition using the Update_Section_Item element

1 Stop the study and make sure that no one can access the study or modify the study 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 MedML Installer utility validates that the CRF RefName and revision is correct, and that the CRF contains the item you updated.

7 Restart the study.

8 Test the updated form version after applying this change. The MedML Installer utility does not validate the item metadata against any existing subject data.

# User

## Purpose

Defines a person who has access to an InForm study database. User access to specific aspects of a study are determined by the user's:

- Site group—Gives the user access to a specific study site.
- Rights group—Gives the user access to a set of rights to perform specific activities.
- Query group—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.
- Signature group—Gives a user the right to sign documents requiring signature. To sign a site's documents, a user must be in a signature group and the appropriate site group.

A user can also be included in the definition of a Sponsor or Site. Inclusion in either of those definitions indicates that the user is a contact at the sponsor or site location.

If you attempt to install information about an existing, active user, including the user's password, the InForm application makes the user inactive as a security precaution.

**Note:** If you use the **IMAGEFILE** attribute, you must load study definition files into the database after you load the resource XML files that define user images.

## 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_DAY"]
   PRODUCTLOCALE="en-US|ja-JP"
   STUDYLOCALE="text"/>
```

### Attributes

**USERNAME**="*name*"

Name that identifies the user in the database. Required.

**USERTYPE**="SYSTEM|SITE|SPONSOR"

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.

Required.

**FIRSTNAME**="*name*"

Given name of the user. Optional.

**LASTNAME**="*name*"

Surname of the user. Optional.

**DISPLAYNAME**="*name*"

User name as displayed in the navigation toolbar in the InForm application. Maximum length is 63 characters; shorter strings are recommended. Optional.

**DESCRIPTION**="*text*"

User description; for example, user's role in the study. Optional.

**TITLE**="*name*"

User title. Optional.

**ADDRESS**="*addr1*"

First line of the user address. Optional.

**ADDRESS2**="*addr2*"

Second line of the user address. Optional.

**CITY**="*name*"

City of the user address. Optional.

**STATE**="*name*"

State of the user address. (Not for use with Province.) Optional.

**PROVINCE**="*name*"

Province of the user address. (Not for use with State.) Optional.

**ZIPCODE**="*code*"

User zip code. (Not for use with Postcode.) Optional.

**POSTCODE**="*code*"

User postal code. (Not for use with Zipcode.) Optional.

**COUNTRY**="*name*"

Country of the user address. Optional.

**PHONE**="*num*"

User telephone number. Optional.

**ALTPHONE**="*num*"

User alternate telephone number. Optional.

**FAX**="*num*"

User fax number. Optional.

**EMAIL**="*addr*"

User email address. Optional.

**HOMESCREENURL**="*url*"

Local or external URL identifying the initial screen that appears when a user logs in to the InForm application. The address must include the http:// prefix and must identify the server on which the file is located by name or by IP address. Optional.

**IMAGEFILE**="*file*"

Name of the image file that appears on the navigation toolbar in the InForm application. Optional.

**IMAGETYPE**="GIF|JPEG|TEXT"

Type of image file: GIF, JPEG, or TEXT. Required if you specified a filename for Imagefile.

**LANGUAGE**="*name*"

User preferred language. English is the default. Optional.

**ACTIVESTATE**="*true*|*false*"

Indicates whether the user is active. The options are true or false. True is the default. Optional.

**DELETESTATE**="*true*|*false*"

Indicates whether the user has been terminated. The options are true or false. False is the default. Terminated users remain in the database. Optional.

**PASSWORD**="*name*"

User password. Optional.

**USERDATEFORMAT**="MONTH_DAY_YEAR|DAY_MONTH_YEAR|YEAR_MONTH_DAY"

Desired date format for viewable InForm application pages for this particular user.

**PRODUCTLOCALE**="*en-US|ja-JP*"

Code for the user product locale. The product locale is the locale in which the system data in the InForm user interface is displayed. The value must be either en-US (English) or ja-JP (Japanese). Required.

**STUDYLOCALE**="*text*"

Code for the study locale of the user. The study locale is the locale in which the study metadata is defined, including visit names, CRF names, section labels, questions, and control labels. The value must match the study locale for a study version in the study. Required.

**UPDATE**="*true|false*"

Perform an incremental update for the specified user by modifying only the properties that you specify.

## Example 1

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="gcp123"
   USERDATEFORMAT="MMM/DD/YYYY"/>
```

## Example 2

The following example adds a user who is located in Switzerland to the database.

```
<USER USERNAME="jcrecy"
   USERTYPE="SITE"
   FIRSTNAME="Jeanne"
   LASTNAME="Crecy"\
   DISPLAYNAME="J Crecy"
   HOMESCREENURL="/inform1/custom/Defaut.html"
   IMAGEFILE="..\Resources\UtilsPhotos\coordinateur_détudes75.gif"
   IMAGETYPE="GIF"
   ACTIVESTATE="false"
   DELETESTATE="false"
   PASSWORD="gcp"/>
   USERDATEFORMAT="DD/MMM/YYYY"
   PRODUCTLOCALE="en-US"
   STUDYLOCALE="fr-CH"/>
```

# UserImage

## Purpose

Allows you to update an image for an existing user. When you use the UserImage element, all other existing user information remains the same.

**Note:** To update other individual user attributes, use the InForm Admin user interface or use the MedML Installer utility. When you update a User 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 application navigation toolbar. If you do not specify a filename, the MedML Installer utility removes the current image file if one is defined for the user. Optional.

**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. English is the default. Optional.

## Example

The following example illustrates the use of the UserImage element to add the image in the xena.jpg file to the user with username "sm."

```
<USERIMAGE USERNAME="sm"
   IMAGEFILE="..\Resources\UserPics\xena.jpg"
   IMAGETYPE="GIF"/>
```

# Userref

## Purpose

Includes previously defined users in the definition of any of the following components:

- QueryGroup
- ReportingGroup
- RightsGroup
- SignatureGroup
- Site
- SiteGroup
- Sponsor

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:** 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 definition.

## Example

The following example illustrates the inclusion of the users Homer and Marge in the definition of a query group called SiteXQueries.

```
<QUERYGROUP GROUPNAME="SiteXQueries">
   <USERREF USERNAME="Homer"/>
   <USERREF USERNAME="Marge"/>
</QUERYGROUP>
```

# VerbatimType

## Purpose

Identifies a specific type of verbatim defined in the coding dictionary.

## Syntax

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

## Attributes

**NAME**="*name*"

Type of verbatim item. 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 WHO-DD dictionary. Optional.

## 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 allows you to define CRF components that you specify as units. You can define unit CRF 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 CRF 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.

### Data object

The Data object allows you to specify how to perform a conversion between the item entered value 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.

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 Windows 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

Allows 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 allows you to pass a message generated by a rule to an email message or to the Windows 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 |
|---|---|
| EMailToGroup | Sends email to the members of a specified query group, signature group, or user manager group defined in the study database. |
| EMailToInFormUser | Sends email to a specified InForm user. |
| EMailToInternetUser | Sends email to any internet email alias. |
| EMailToUser | For internal use only. |
| GetNamedValue | Returns the data value associated with a name specified in the AddNamedValue method on the Patient or Result object. |
| LogMessage | Logs the specified message to the Windows log. |
| OutputDebug | Not supported. |

## Site object

Allows you to retrieve information about the current site. The Site object has the following properties, which are Get only:

| Property | Type | What the property holds |
|---|---|---|
| ContactID | DWORD | Database ID of the InForm user indicated as the primary contact for the site. |
| SiteName | string | Site name. |
| Address | string | Site address. |
| Phone | string | Site phone number. |
| AltPhone | string | Alternate site phone number. |
| Beeper | string | Site beeper number. |
| Email | string | Site email address. |
| Fax | string | Site fax number. |
| TimeZone | string | Site time zone. |

### Site method

| Method | Purpose |
|---|---|
| SetSite | Loads the Site object from cache with data from the site with the specified site ID. For Oracle use only. |

## User object

Allows you to retrieve information about the current user. The User object has the following properties, which are Get only:

| Property | Type | What the property holds |
|---|---|---|
| FirstName | string | User given name. |
| LastName | string | User surname. |
| Address | string | User address. |
| Phone | string | User phone number. |
| AltPhone | string | Alternate user phone number. |
| Beeper | string | User beeper number. |
| Email | string | User email address. |
| Fax | string | User fax number. |

| Property | Type | What the property holds |
|---|---|---|
| UserName | string | User account name used to log in to the InForm application. |

### User method

| Method | Purpose |
|---|---|
| SetUser | Loads the User object from cache with data from the user with the specified user ID. For Oracle use only. |

## EMailToGroup(GroupID,Subject,Msg)

EMailToGroup is for internal Oracle use only.

Sends email to the members of a specified query group, signature group, or user manager group defined in the study 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/OracleHS/InForm/PFMngrExecutionPlan. The entry is a string value named "FromAddress". The default is "nobody@Oracle.com".

## EMailToInFormUser "InFormUser,szSubject,szMsg"

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/OracleHS/InForm/PFMngrExecutionPlan. The entry is a string value named "FromAddress". The default is "nobody@Oracle.com".

### Example

Global.EMailToInformUser "sm", "CRFUnLock", "Sending InForm User email when Unlock-CRF"

## EMailToInternetUser "EmailAddress,szSubject,szMsg"

Sends email to any email alias.

### Arguments

**EmailAddress**

The 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/OracleHS/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)

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@phaseforward.com","SAE Occurrence",emailtxt)
end if
```

## LogMessage "Message"

Logs the specified message to the Windows log.

### Argument

**Message**

String containing the text of the message to log to the Windows log.

### Example

```
Global.LogMessage "Query generated"
```

## SetSite(SiteID)

SetSite(SiteID) is for internal use only.

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.

Loads the User object from cache with data from the user with the specified user ID.

#### Argument

**UserID**

Database IF for the user you want to access by using the User object.

## Randomization objects and methods

When you implement randomization in a study, you must perform several configuration steps. One of these is to create a script that generates a drug kit number to assign to each subject as the subject is enrolled. This randomization script can use the objects and methods that are available for rules and calculations, and can also use the Randomization object and method described in this topic.

Use MedML and the MedML Installer utility to attach the randomization script to a calculated control on the CRF where you want to generate the randomization sequence number and assign a drug kit. A predefined section definition called DRUGKITSECTION is provided in the core_randomization.xml file in the XMLBase folder in the InForm installation tree.

### Randomization object

Allows you to assign the subject, site, randomization, and study 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 subject. |
| 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 | Holds the randomization type:<br>• 1 (Simple Central)—The study uses one list of drug kits. Each new subject is assigned the next sequential drug kit number on the list.<br>• 2 (Central Stratified)—The study has multiple lists of drug kits. Each new subject is assigned to a drug kit list based on entered subject data. Then, the subject is assigned the next sequential drug kit number on that list.<br>• 3 (Simple Site)—Each site has a different drug kit list. Each new subject is assigned the next sequential drug kit number on the list for the subject's site.<br>• 4 (Stratified by Site)—Each site has multiple lists of drug kits. Each new subject is first assigned to the set of list for the subject's site. Then, the subjectis assigned to one of the site's drug kit lists based on entered subject data. Finally, the subject is assigned the next sequential drug kit number on that list. |
| Source | Randomization source list name, or stratification code. |

| Property | Purpose |
|---|---|
| Revision | Revision number of the study version. |

## GetNextKit method (KitInfo)

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 subject's weight, as entered in the Demographics form in Visit 1. Based on the subject's weight, the InForm software 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 subject's site determines the randomization source list from which the InForm application gets the next sequence number.

```
Function GetRndSourceList()
szSite = Patient.GetSiteName();
IF (szSite = "Mass General") 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 subject'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 "Mass General"
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)

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 CRF rules (scripts that perform edit checks on data entered into an item on a CRF), and calculations (scripts that compute the value of a data item on a CRF based on the value of one or more related items) can use the following objects.

**Note:** You can create CRF rule and calculation definitions either by using the Rule Wizard in the Central Designer application, or by using MedML and the MedML Installer utility.

### Patient object

Allows you to obtain form values that pertain to a subject. 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 CRF 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 CRF rules and calculations can use many of the same methods. Except where noted, Patient methods are available for both CRF rules and calculations.

| Method | Purpose |
|---|---|
| AddNamedValue | Adds a name value pair to the list of arguments that the GetNamedValue method can access. |
| ClearValues | Clears the properties of the specified object. |
| FormHasState | Indicates whether a CRF is in a state that you specify. FormHasState returns a Boolean value. |
| FormHasStateEx | Returns a Boolean value indicating whether a CRF is in a specified state. |
| FormHasStateRF | Returns a Boolean value indicating whether an instance of a repeating CRF is in a specified state. |
| GetArgument | Returns the value of the specified rule argument. |
| GetArgumentCount | Returns the number of arguments used in the rule context. |
| GetAssociationCount | Returns the number of repeating form instances that have been associated with a specific instance of a CRF within a visit. |
| GetAssociationValue | Iterates over the associations of a CRF 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. |
| GetCurPath | Returns the path of the context in which the rule is running. |
| GetCurrentFormIndex | Returns a number containing the index of the current instance of a repeating form. |
| GetCurrentISRowNum | Returns a number containing the index of the current row or 0 if it is a new row. |
| GetCurrentVisitIndex | Returns a number containing the index of the current visit. |
| GetDefaultValue | Is the same as GetValue, with default parameters. |
| GetEnteredValue | Returns the value of the data entered in the specified CRF component. |
| GetEnteredValueRF | Returns the value of the data entered in the specified CRF component in a specific instance of a repeating CRF. |
| GetFormCount | Returns the returns the number of repeating CRF instances that have been created for a specified visit. |
| GetID | Returns the internal database ID of the current subject. |
| GetItemsetCount | Returns the number of rows currently entered in an itemset.<br>**Note:** Do not use this as a means to sequentially number or identify a row. |

| Method | Purpose |
|---|---|
| GetItemsetRowID | Returns the uniquely identifying row ID of an itemset row. |
| GetItemsetCountRF | Returns the number of rows currently entered in an itemset in a specific instance of a repeating CRF. |
| GetPatientNumber | Returns the number assigned to the current subject. |
| GetRefName | Returns a string containing the RefName of the specified object type in the context in which the rule is run. |
| GetRefPath | Returns a string containing the RefName path of the specified object type in the context in which the rule is run. |
| GetScreeningNumber | Returns a string containing the current subject's screening number. |
| GetServerTime | Returns a string containing the datetime value of the current server as GMT. |
| GetSiteID | Returns the internal ID of the current subject's site. |
| GetSiteMnemonic | Returns a string containing the site mnemonic for the current subject's site. |
| GetSiteName | Returns the name of the current subject's site. |
| GetSiteTime | Returns the datetime value of the server in the current site's time zone. |
| GetTrialName | Returns the name of the study in which the current subject is participating. |
| GetValue | Returns the normalized value of the data entered in the specified CRF component. |
| GetValueRF | Returns the normalized value of the data entered in the specified CRF component in a specific instance of a repeating CRF. |
| GetVisitCount | Returns the number of visits for the visit specified in VisitRefName. |
| IsCBSelected | Checks to see if a simple control in a checkbox control is selected and returns a Boolean value containing the state of the checkbox. |
| IsCBSelectedRF | In a specific instance of a repeating CRF, checks to see if a simple control in a checkbox control is selected and returns a Boolean value containing the state of the checkbox. |
| IsItemSetDeleted | Indicates whether an item in an add entry itemset is deleted. |
| IsItemSetDeletedRF | Indicates whether an item in an add entry itemset is deleted in a specific instance of a repeating CRF. |
| IsValidItemPath | Confirms whether a RefName path is valid, by checking the database for the existence of the specified path. |
| IsValidItemPathRF | In a specific instance of a repeating CRF, confirms whether a RefName path is valid, by checking the database for the existence of the specified path. |
| ItemHasBeenNormalizedEx | Indicates whether the units associated with the referenced data item have been selected, by determining whether the data has been normalized. |

| Method | Purpose |
|---|---|
| ItemHasBeenNormalizedRF | In a specific instance of a repeating CRF, indicates whether the units associated with the referenced data item have been selected, by determining whether the data has been normalized. |
| LoadCRB | Loads the subject case book into memory. |
| OutputDebug | Displays the value of a specified variable as a test script runs. |
| RemoveEmptyDynamicVisit | Removes a dynamic visit that a patient has been scheduled into, if no data has been entered into any of the visit's CRFs. |
| RemoveNotStartedDynamicForm | Removes a dynamic form from a subject's case book, provided the CRF has not been started. |
| SetDynamicVisitStart | Schedules a subject into a dynamic visit. |
| SetEnteredValue | Uses the value of the data entered in the specified CRF component. |
| SetEnteredValueRF | In a specific instance of a repeating CRF, uses the value of the data entered in the specified CRF component. |
| SetNewValue | For internal Oracle use only. |
| SetNewValueRF | For internal Oracle use only. |
| SetValue | Uses the normalized value of the data entered in the specified CRF component. |
| SetValueRF | In a specific instance of a repeating CRF, uses the normalized value of the data entered in the specified CRF component. |
| ShowDynamicForm | Inserts a dynamic CRF into a visit based on the outcome of a calculation on another CRF. |

## Result object

Allows you to get or set the result of an edit check generated by a CRF 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.

| 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 CRF 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. |

### Result methods

The methods available for use with the Result object are the same methods as are available for the Patient object.

| Method | Purpose |
|---|---|
| AddNamedValue | Adds a name value pair to the list of arguments that the GetNamedValue method can access. |
| ClearValues | Clears the properties of the specified object. |
| FormHasState | Indicates whether a CRF is in a state that you specify. FormHasState returns a Boolean value. |
| FormHasStateEx | Returns a Boolean value indicating whether a CRF is in a specified state. |
| FormHasStateRF | Returns a Boolean value indicating whether an instance of a repeating CRF is in a specified state. |
| GetArgument | Returns the value of the specified rule argument. |
| GetArgumentCount | Returns the number of arguments used in the rule context. |
| GetAssociationCount | Returns the number of repeating form instances that have been associated with a specific instance of a CRF within a visit. |
| GetAssociationValue | Iterates over the associations of a CRF 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. |
| GetCurPath | Returns the path of the context in which the rule is running. |
| GetCurrentFormIndex | Returns a number containing the index of the current instance of a repeating form. |
| GetCurrentISRowNum | Returns a number containing the index of the current row or 0 if it is a new row. |
| GetCurrentVisitIndex | Returns a number containing the index of the current visit. |
| GetDefaultValue | Is the same as GetValue, with default parameters. |
| GetEnteredValue | Returns the value of the data entered in the specified CRF component. |
| GetEnteredValueRF | Returns the value of the data entered in the specified CRF component in a specific instance of a repeating CRF. |
| GetFormCount | Returns the returns the number of repeating CRF instances that have been created for a specified visit. |
| GetID | Returns the internal database ID of the current subject. |
| GetItemsetCount | Returns the number of rows currently entered in an itemset.<br><br>**Note:** Do not use this as a means to sequentially number or identify a row. |

| Method | Purpose |
|---|---|
| GetItemsetRowID(REFNAMEPath,VisitIndex) | Returns the uniquely identifying row ID of an itemset row. |
| GetItemsetCountRF | Returns the number of rows currently entered in an itemset in a specific instance of a repeating CRF. |
| GetPatientNumber | Returns the number assigned to the current subject. |
| GetRefName | Returns a string containing the RefName of the specified object type in the context in which the rule is run. |
| GetRefPath | Returns a string containing the RefName path of the specified object type in the context in which the rule is run. |
| GetScreeningNumber | Returns a string containing the current subject's screening number. |
| GetServerTime | Returns a string containing the datetime value of the current server as GMT. |
| GetSiteID | Returns the internal ID of the current subject's site. |
| GetSiteMnemonic | Returns a string containing the site mnemonic for the current subject's site. |
| GetSiteName | Returns the name of the current subject's site. |
| GetSiteTime | Returns the datetime value of the server in the current site's time zone. |
| GetTrialName | Returns the name of the study in which the current subject is participating. |
| GetValue | Returns the normalized value of the data entered in the specified CRF component. |
| GetValueRF | Returns the normalized value of the data entered in the specified CRF component in a specific instance of a repeating CRF. |
| GetVisitCount | Returns the number of visits for the visit specified in VisitRefName. |
| IsCBSelected | Checks to see if a simple control in a checkbox control is selected and returns a Boolean value containing the state of the checkbox. |
| IsCBSelectedRF | In a specific instance of a repeating CRF, checks to see if a simple control in a checkbox control is selected and returns a Boolean value containing the state of the checkbox. |
| IsItemSetDeleted | Indicates whether an item in an add entry itemset is deleted. |
| IsItemSetDeletedRF | Indicates whether an item in an add entry itemset is deleted in a specific instance of a repeating CRF. |
| IsValidItemPath | Confirms whether a RefName path is valid, by checking the database for the existence of the specified path. |
| IsValidItemPathRF | In a specific instance of a repeating CRF, confirms whether a RefName path is valid, by checking the database for the existence of the specified path. |

| Method | Purpose |
|---|---|
| ItemHasBeenNormalizedEx | Indicates whether the units associated with the referenced data item have been selected, by determining whether the data has been normalized. |
| ItemHasBeenNormalizedRF | In a specific instance of a repeating CRF, indicates whether the units associated with the referenced data item have been selected, by determining whether the data has been normalized. |
| LoadCRB | Loads the subject case book into memory. |
| OutputDebug | Displays the value of a specified variable as a test script runs. |
| RemoveEmptyDynamicVisit | Removes a dynamic visit that a patient has been scheduled into, if no data has been entered into any of the visit's CRFs. |
| RemoveNotStartedDynamicForm | Removes a dynamic form from a subject's case book, provided the CRF has not been started. |
| SetDynamicVisitStart | Schedules a subject into a dynamic visit. |
| SetEnteredValue | Uses the value of the data entered in the specified CRF component. |
| SetEnteredValueRF | In a specific instance of a repeating CRF, uses the value of the data entered in the specified CRF component. |
| SetNewValue | For internal Oracle use only. |
| SetNewValueRF | For internal Oracle use only. |
| SetValue | Uses the normalized value of the data entered in the specified CRF component. |
| SetValueRF | In a specific instance of a repeating CRF, uses the normalized value of the data entered in the specified CRF component. |
| ShowDynamicForm | Inserts a dynamic CRF into a visit based on the outcome of a calculation on another CRF. |

## AddNamedValue"Name","Value"

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 are case-insensitive.

### 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()

Clears the properties and name value pairs of the specified object.

## FormHasState(Visit,Form,State)

Indicates whether a CRF 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.

### Arguments

**Visit**

RefName of the visit.

> **Note:** This is the RefName only of the visit, not the full REFNAMEPath.

**Form**

RefName of the CRF.

> **Note:** This is the RefName only of the CRF, not the full REFNAMEPath.

**State**

String specifying the state you want to check. The following states are valid:

- **Frozen**—CRF is frozen.
- **Hascomment**—CRF has a comment at the CRF-level.
- **Hasdata**—CRF has data.
- **Locked**—CRF is locked.
- **Missing**—CRF has missing data.
- **Queries**—CRF has queries.
- **SDVComplete**—CRF is marked Verified.
- **SDVPartial**—CRF Is marked Not Complete.
- **SDVReady**—CRF has been marked ready for source verification.
- **Signed**—CRF has been signed.
- **Skipped**—CRF has been marked not completed with a CRF-level comment.
- **Started**—CRF was started with subject 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)

Indicates whether a CRF is in a state that you specify. FormHasStateEx returns a Boolean value. FormHasStateEx is exactly the same as FormHasState except that FormHasState Ex includes the VisitIndex argument so that you can target a specific instance of a repeating visit.

### Arguments

**Visit**

RefName of the visit.

> **Note:** This is the RefName only of the visit, not the full REFNAMEPath.

**VisitIndex**

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 CRF. Note that this is the RefName only of the CRF, not the full REFNAMEPath.

**State**

String specifying the state you want to check. The following states are valid:

- **Frozen**—CRF is frozen.
- **Hascomment**—CRF has a comment at the CRF-level.
- **Hasdata**—CRF has data.
- **Locked**—CRF is locked.
- **Missing**—CRF has missing data.
- **Queries**—CRF has queries.
- **SDVComplete**—CRF is marked Verified.
- **SDVPartial**—CRF Is marked Not Complete.
- **SDVReady**—CRF has been marked ready for source verification.
- **Signed**—CRF has been signed.
- **Skipped**—CRF has been marked not completed with a CRF-level comment.
- **Started**—CRF was started with subject data, comments, or queries.

## Example

For example, if an AE CRF in the second instance of an unscheduled visit indicates a concomitant medication was taken, verify the ConMed CRF 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 CRFs, the FormHasStateRF method indicates whether a CRF is in a state that you specify. FormHasStateRF returns a Boolean value. FormHasStateRF is exactly the same as FormHasStateEx except that FormHasStateRF:

- Includes the FormIndex argument so that you can target a specific instance of a repeating CRF.
- Allows 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**

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**

Index into a repeating CRF.

- **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**—CRF is frozen.
- **Hascomment**—CRF has a comment at the CRF-level.
- **Hasdata**—CRF has data.
- **Locked**—CRF is locked.
- **Missing**—CRF has missing data.
- **Queries**—CRF has queries.
- **SDVComplete**—CRF is marked Verified.
- **SDVPartial**—CRF Is marked Not Complete.
- **SDVReady**—CRF has been marked ready for source verification.
- **Signed**—CRF has been signed.

- **Skipped**—CRF has been marked not completed with a CRF-level comment.
- **Started**—CRF was started with subject data, comments, or queries.
- **Deleted (for instances of repeating CRFs)**—Indicates that the CRF 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

Returns the value of the specified rule argument. Arguments are defined with RULEARG elements 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()

Returns the number of arguments used in the rule context.

## GetAssociationCount(RefNamePath,VisitIndex,FormIndex)

Returns the number of repeating CRF instances that have been associated with a specific instance of a CRF within a visit.

### Arguments

**RefNamePath**

String that identifies the CRF to get. Each RefName in the string is the RefName specified in the XML file that contains the definition of the CRF component. The RefNamePath must be fully qualified; only the subject component of the path defaults to the current subject. The RefNamePath argument is in the following format: *(0.Visit.Form)*

- **0**—Current subject.
- **Visit**—RefName of the visit, as specified in the XML file that contains the visit definition.
- **Form**—RefName of the CRF, as specified in the XML file that contains the CRF definition.

**VisitIndex**

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**

Index into a repeating CRF. CRF indexes start with 1 for the first instance of the CRF.

### 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 CRF. GetAssociationCount works in conjunction with the GetAssociationValue 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)

Works in conjunction with the GetAssociationCount method to return information about a specific associated CRF instance. GetAssociationValue iterates over the associations of a CRF 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 CRF
- VisitIndex
- FormIndex

### Arguments

**nAssociations**

Number of instances of the associated CRF 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()

Returns the RefName path of the context in which the rule is running.

### Example

```
val = Patient.GetValue(Patient.GetCurPath(), "", 0,0,0)
```

## GetCurrentFormIndex()

Returns 0 or a number containing the index of the current instance of a repeating CRF. Use this method to determine which instance of a repeating CRF is being changed, or whether a new instance is being added on Submit.

During the first submit of a repeating CRF 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 CRF to the GetAssociationCount 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()

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 CRF, captures information about the Adverse Experience being entered into the CRF. 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 "passsitemnemonic", sitemnemonic
patient.AddNamedValue "passscreeningnumber", 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("passsitemnemonic"))
   getscreeningnumber = cstr(global.GetNamedValue("passscreeningnumber"))
   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& ",
      Subject Number = " &getpatientnumber
  global.EMailToInternetUser "user@Oracle.com", emailsub , emailtxt
end if
```

## GetCurrentVisitIndex()

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 same as the GetValue method, with default parameters. GetDefaultValue assumes the following default values for AttributeName, Index, ItemsetIndex, and VisitIndex:

- **AttributeName="VALUE"**—Method returns the value of the CRF component.
- **Index="0"**—CRF 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 CRF component. If the item is not normalized, the method returns the entered value.

### Arguments

**RefNamePath**

String that identifies the CRF component to get. Each RefName in the string is the RefName specified in the XML file that contains the definition of the CRF component. The REFNAMEPath must be fully qualified; only the subject component of the path defaults to the current subject. The REFNAMEPath argument is in the following format:
(0.Visit.Form.Section.Itemset.Item[.control[.control...]])

- **0**—Current subject.
- **Visit**—RefName of the visit, as specified in the XML file that contains the visit definition.
- **Form**—RefName of the CRF, as specified in the XML file that contains the CRF 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 subject 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 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.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)

Returns the value of the data entered in the specified CRF component. GetEnteredValue is very similar to the GetValue method, which returns the normalized value of the data entered in the specified CRF component. The difference between these methods applies to form components that are defined with units.

The MedML Installer utility allows you to define CRF components that you specify as units. You can define unit CRF 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 CRF 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.

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:** To check a data value on a repeating CRF, or to attach a rule to a repeating form in order to reference a value on a non-repeating CRF, use GetValueRF or GetEnteredValueRF.

### Arguments

**RefNamePath**

String that identifies the CRF component to get. Each RefName in the string is the RefName specified in the XML file that contains the definition of the CRF component. The REFNAMEPath must be fully qualified; only the subject component of the path defaults to the current subject. The REFNAMEPath argument is in the following format:
(0.Visit.Form.Section.Itemset.Item[.control[.control...]])

- **0**—Current subject.
- **Visit**—RefName of the visit, as specified in the XML file that contains the visit definition.
- **Form**—RefName of the CRF, as specified in the XML file that contains the CRF 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 subject 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 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.TextControlRefname*.

**AttributeName**

String that specifies the type of information to return, specified in double quotes (""). Valid AttributeNames are:

- **VALUE**—Gets the value of the CRF component. This attribute is the default; you can indicate it with empty double quotes.
- **DATESTRING**—Returns only the date portion of a DateTime control.
- **TIMESTRING**—Returns only the time portion of a DateTime control.
- **DATETIMESTRING**—Returns both the date and the time portions of a DateTime control.
- **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.

  **Note: The SYMBOL attribute of the unit can be translated. The value that is returned is in the locale of the current subject site. Therefore, any rule logic that checks the returned symbol text must take the locale of the current subject site into consideration.**

- **VALIDDATEMAP**—Number that specifies whether a date is complete. The script uses the **VALIDDATEMAP** attribute of the GetEnteredValue method on the Patient object. **VALIDDATEMAP** returns an integer that contains a bitmask indicating which elements of the DateTime control 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 GetEnteredValue method on the Patient object. **UNKNOWNDATEMAP** returns an integer that contains a bitmask indicating which elements of the DateTime control 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**

Starting with 0, an index into an array of the selected items in a multiple-value control such as a list of checkboxes. 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**

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 subject 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 clicks 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**

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 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 CRF. 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 BPREADINGGROUP group of the BPREADINGITEM 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 CRF. 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, returns the value of the data entered in the specified CRF component. The GetEnteredValueRF method is identical to the GetEnteredValue method, except that it is intended for use in repeating CRFs. Therefore, it has an additional argument, FormIndex, which allows you to reference a specific instance of the repeating CRF.

**Note:** Use GetEnteredValueRF either when you check a value on a repeating CRF or when you attach a rule to a repeating CRF and reference a value in a non-repeating CRF.

### Arguments

**RefNamePath**

String that identifies the CRF component to get. Each RefName in the string is the RefName specified in the XML file that contains the definition of the CRF component. The REFNAMEPath must be fully qualified; only the subject component of the path defaults to the current subject. The REFNAMEPath argument is in the following format:
(0.Visit.Form.Section.Itemset.Item[.control[.control...]])

- **0**—Current subject.
- **Visit**—RefName of the visit, as specified in the XML file that contains the visit definition.
- **Form**—RefName of the CRF, as specified in the XML file that contains the CRF 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 subject 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 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.TextControlRefname*.

**AttributeName**

String that specifies the type of information to return, specified in double quotes (""). Valid AttributeNames are:

- **VALUE**—Gets the value of the CRF component. This attribute is the default; you can indicate it with empty double quotes.
- **DATESTRING**—Returns only the date portion of a DateTime control.
- **TIMESTRING**—Returns only the time portion of a DateTime control.
- **DATETIMESTRING**—Returns both the date and the time portions of a DateTime control.
- **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.

  **Note: The SYMBOL attribute of the unit can be translated. The value that is returned is in the locale of the current subject site. Therefore, any rule logic that checks the returned symbol text must take the locale of the current subject site into consideration.**

- **VALIDDATEMAP**—Number that specifies whether a date is complete. The script uses the **VALIDDATEMAP** attribute of the GetEnteredValue method on the Patient object. **VALIDDATEMAP** returns an integer that contains a bitmask indicating which elements of the DateTime control 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 GetEnteredValue method on the Patient object. **UNKNOWNDATEMAP** returns an integer that contains a bitmask indicating which elements of the DateTime control 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**

Starting with 0, an index into an array of the selected items in a multiple-value control such as a list of checkboxes. 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**

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 subject 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 clicks 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**

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**

Index into a repeating CRF.

- **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 CRF, 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)

Returns the number of repeating CRF instances that have been created for a specified visit.

### Arguments

**Visit**

RefName of the formset in which the repeating CRF occurs.

**VisitIndex**

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 CRF.

### 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()

Returns the ID of the current subject. The ID is an internally assigned value by which the subject 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)

Returns the number of rows currently entered in an itemset.

### Arguments

**RefNamePath**

String that identifies the CRF component to get. Each RefName in the string is the RefName specified in the XML file that contains the definition of the CRF component. The REFNAMEPath must be fully qualified; only the subject component of the path defaults to the current subject. The REFNAMEPath argument is in the following format:
(0.Visit.Form.Section.Itemset.Item[.control[.control...]])

- **0**—Current subject.
- **Visit**—RefName of the visit, as specified in the XML file that contains the visit definition.
- **Form**—RefName of the CRF, as specified in the XML file that contains the CRF 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 subject 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 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.TextControlRefname*.

**VisitIndex**

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)

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 CRF component to get. Each RefName in the string is the RefName specified in the XML file that contains the definition of the CRF component. The REFNAMEPath must be fully qualified; only the subject component of the path defaults to the current subject. The REFNAMEPath argument is in the following format:
(0.Visit.Form.Section.Itemset.Item[.control[.control...]])

- **0**—Current subject.
- **Visit**—RefName of the visit, as specified in the XML file that contains the visit definition.
- **Form**—RefName of the CRF, as specified in the XML file that contains the CRF 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 subject 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 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.TextControlRefname*.

**VisitIndex**

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**

Index into a repeating CRF.

- **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 CRF, which is repeating.

```
CheckItem = "0.Visit2.AE.AE.AE.AETEXTITEM"
CheckCount = Cint(Patient.GetItemsetCountRF(CheckItem,1,1))
```

## GetItemsetRowID(REFNAMEPath,VisitIndex)

Returns the unique row ID of the itemset row.

### Arguments

**RefNamePath**

String that identifies the CRF component to get. Each RefName in the string is the RefName specified in the XML file that contains the definition of the CRF component. The REFNAMEPath must be fully qualified; only the subject component of the path defaults to the current subject. The REFNAMEPath argument is in the following format:
(0.Visit.Form.Section.Itemset.Item[.control[.control...]])

- **0**—Current subject.
- **Visit**—RefName of the visit, as specified in the XML file that contains the visit definition.
- **Form**—RefName of the CRF, as specified in the XML file that contains the CRF 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 subject 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 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.TextControlRefname*.

**VisitIndex**

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()

Returns the number assigned to the current subject. The subject number is a value assigned to a subject during enrollment.

### Example

See the example for GetCurrentISRowNum()

## GetRefName(ObjType)

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)

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()

Returns a string containing the current subject screening number.

### Example

See the example for GetCurrentISRowNum()

## GetServerTime()

Returns a string containing the datetime value of the current server as GMT.

### Example

See the example for GetCurrentISRowNum()

## GetSiteID()

Returns the current subject site ID. The site ID is an internally assigned value by which the site is known in the database.

### Example

```
SiteID = Patient.GetSiteID()
If SiteID...
```

## GetSiteLocale()

Returns the study locale for the site for the current subject; for example, **en-US,** or **ja-JP**.

### Example

See the example for GetSiteID().

## GetSiteMnemonic()

Returns a string containing the site mnemonic for the current subject site. This method, which is similar to the GetSiteName method, provides an abbreviated way to identify a site.

### Example

See the example for GetCurrentISRowNum()

## GetSiteName()

Returns the name of the current subject site.

### Example

```
SiteName = Patient.GetSiteName()
If SiteName...
```

## GetSiteTime()

Returns the datetime value of the server in the current site time zone. This method is useful for ensuring that a future date or time is not entered on a CRF.

### Example

See the example for GetCurrentISRowNum()

## GetTrialName()

Returns the name of the subject in which the current subject is participating.

### Example

```
TrialName = Patient.GetTrialName()
If TrialName...
```

## GetUserName()

Returns the name of user who is currently logged in to the InForm application.

### Example

See the example for GetSiteID().

## GetValue(REFNAMEPath,AttributeName,Index,ItemsetIndex,VisitIndex)

Returns the normalized value of the data entered in the specified CRF component. If the item is not normalized, the method returns the entered value. GetValue is very similar to the GetEnteredValue method, which returns the entered value of the data entered in the specified CRF component.

The difference between these methods applies to CRF components that are defined as units.

The MedML Installer utility allows you to define CRF components that you specify as units. You can define unit CRF 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 CRF 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.

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 CRF, or to attach a rule to a repeating CRF in order to reference a value on a non-repeating CRF, use GetValueRF or GetEnteredValueRF

### Arguments

**RefNamePath**

String that identifies the CRF component to get. Each RefName in the string is the RefName specified in the XML file that contains the definition of the CRF component. The REFNAMEPath

must be fully qualified; only the subject component of the path defaults to the current subject. The REFNAMEPath argument is in the following format:
(0.Visit.Form.Section.Itemset.Item[.control[.control...]])

- **0**—Current subject.
- **Visit**—RefName of the visit, as specified in the XML file that contains the visit definition.
- **Form**—RefName of the CRF, as specified in the XML file that contains the CRF 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 subject 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 CRF component. This attribute is the default; you can indicate it with empty double quotes.
- **DATESTRING**—Returns only the date portion of a DateTime control.
- **TIMESTRING**—Returns only the time portion of a DateTime control.
- **DATETIMESTRING**—Returns both the date and the time portions of a DateTime control.
- **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.

  **Note: The SYMBOL attribute of the unit can be translated. The value that is returned is in the locale of the current subject site. Therefore, any rule logic that checks the returned symbol text must take the locale of the current subject site into consideration.**

- **VALIDDATEMAP**—Number that specifies whether a date is complete. The script uses the **VALIDDATEMAP** attribute of the GetEnteredValue method on the Patient object. **VALIDDATEMAP** returns an integer that contains a bitmask indicating which elements of the DateTime control 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 GetEnteredValue method on the Patient object. **UNKNOWNDATEMAP** returns an integer that contains a bitmask indicating which elements of the DateTime control 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**

Starting with 0, an index into an array of the selected items in a multiple-value control such as a list of checkboxes. 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**

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 subject 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 clicks 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**

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.

## GetValueRF(REFNAMEPath,AttributeName,Index,ItemsetIndex,VisitIndex,FormIndex)

In instances of a repeating CRF, returns the normalized value of the data entered in the specified CRF component. If the item is not normalized, the method returns the entered value. The GetValueRF method is identical to the GetValue method, except that it is intended for use in repeating CRFs. Therefore, it has an additional argument, FormIndex, which enables you to reference a specific instance of the repeating CRF.

**Note:** Use GetValueRF either when you check a value on a repeating CRF or when you attach a rule to a repeating CRF and reference a value in a non-repeating CRF.

## Arguments

**RefNamePath**

String that identifies the CRF component to get. Each RefName in the string is the RefName specified in the XML file that contains the definition of the CRF component. The REFNAMEPath must be fully qualified; only the subject component of the path defaults to the current subject. The REFNAMEPath argument is in the following format:
(0.Visit.Form.Section.Itemset.Item[.control[.control...]])

- **0**—Current subject.
- **Visit**—RefName of the visit, as specified in the XML file that contains the visit definition.
- **Form**—RefName of the CRF, as specified in the XML file that contains the CRF 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 subject 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 CRF component. This attribute is the default; you can indicate it with empty double quotes.
- **DATESTRING**—Returns only the date portion of a DateTime control.
- **TIMESTRING**—Returns only the time portion of a DateTime control.
- **DATETIMESTRING**—Returns both the date and the time portions of a DateTime control.
- **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.

  **Note: The SYMBOL attribute of the unit can be translated. The value that is returned is in the locale of the current subject site. Therefore, any rule logic that checks the returned symbol text must take the locale of the current subject site into consideration.**

- **VALIDDATEMAP**—Number that specifies whether a date is complete. The script uses the

**VALIDDATEMAP** attribute of the GetEnteredValue method on the Patient object. **VALIDDATEMAP** returns an integer that contains a bitmask indicating which elements of the DateTime control 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 GetEnteredValue method on the Patient object. **UNKNOWNDATEMAP** returns an integer that contains a bitmask indicating which elements of the DateTime control 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**

Starting with 0, an index into an array of the selected items in a multiple-value control such as a list of checkboxes. 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**

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 subject 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 clicks 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**

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**

Index into a repeating CRF.

- **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 CRF instance and in each succeeding CRF 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,"",0,0,0,1)
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 forthat's
End If
End If
Next
```

## GetVisitCount(VisitRefName)

Returns the number of instances for the visit specified in the **VisitRefName** argument. This method is used for repeating visits, and can be used to establish a constant for counting iterations through a loop of all repetitions of a visit.

### Arguments

**VisitRefName**

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))
```

## IsCBSelected(RefNamePath,ItemSetIndex, VisitIndex)

Checks to see if a child control in a checkbox control is selected and returns a Boolean value containing the state of the child control.

- **True**—Selected
- **False**—Deselected

### Arguments

**RefNamePath**

String that identifies the CRF component to get. Each RefName in the string is the RefName specified in the XML file that contains the definition of the CRF component. The REFNAMEPath must be fully qualified; only the subject component of the path defaults to the current subject. The REFNAMEPath argument is in the following format:
(0.Visit.Form.Section.Itemset.Item[.control[.control...]])

- **0**—Current subject.
- **Visit**—RefName of the visit, as specified in the XML file that contains the visit definition.
- **Form**—RefName of the CRF, as specified in the XML file that contains the CRF 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 subject 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 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.TextControlRefname*.

**ItemSetIndex**

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 subject 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 clicks 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**

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 simple control in a checkbox control to determine whether it is selected. If all checkboxes are empty, it generates a query if the radio button for the checkbox control 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 CRF, checks to see if a child control in a checkbox control is selected and returns a Boolean value containing the state of the child control.

- **True**—Selected
- **False**—Deselected

The IsCBSelectedRF method is identical to the IsCBSelected method, except that it is intended for use in repeating CRFs. Therefore, it has an additional argument, **FormIndex**, which allows you to reference a specific instance of the repeating CRF.

### Arguments

**RefNamePath**

String that identifies the CRF component to get. Each RefName in the string is the RefName specified in the XML file that contains the definition of the CRF component. The REFNAMEPath must be fully qualified; only the subject component of the path defaults to the current subject. The REFNAMEPath argument is in the following format:
(0.Visit.Form.Section.Itemset.Item[.control[.control...]])

- **0**—Current subject.
- **Visit**—RefName of the visit, as specified in the XML file that contains the visit definition.
- **Form**—RefName of the CRF, as specified in the XML file that contains the CRF 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 subject 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 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.TextControlRefname*.

**ItemSetIndex**

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 subject 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 clicks 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**

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**

Index into a repeating CRF.

- **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 checkbox in the Mthd_GC control of the current instance of the CM CRF to determine whether it has been selected. If no checkboxes have been selected, the rule issues a query.

```
patient.RulePassed=1
If patient.GetValueRF("0.CommonCRF.CM.sctCM.0.itmCMMed.txtCMMed","",0,0,0,0)<>"" then
If ("True" = patient.IsCBSelectedRF
("0.CommonCRF.CM.sctCM.0.itmMthd.Mthd_GC.Mthd_l_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)

Indicates whether a row in an Add Entry 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 CRF component to get. Each RefName in the string is the RefName specified in the XML file that contains the definition of the CRF component. The REFNAMEPath must be fully qualified; only the subject component of the path defaults to the current subject. The REFNAMEPath argument is in the following format:
(0.Visit.Form.Section.Itemset.Item[.control[.control...]])

- **0**—Current subject.
- **Visit**—RefName of the visit, as specified in the XML file that contains the visit definition.
- **Form**—RefName of the CRF, as specified in the XML file that contains the CRF 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 subject 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 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.TextControlRefname*.

**VisitIndex**

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**

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 subject 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 clicks 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 CRF, indicates whether a row in an Add Entry 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 CRFs. Therefore, it has an additional argument, FormIndex, which allows you to reference a specific instance of the repeating CRF.

### Arguments

**RefNamePath**

String that identifies the CRF component to get. Each RefName in the string is the RefName specified in the XML file that contains the definition of the CRF component. The REFNAMEPath must be fully qualified; only the subject component of the path defaults to the current subject. The REFNAMEPath argument is in the following format:
(0.Visit.Form.Section.Itemset.Item[.control[.control...]])

- **0**—Current subject.
- **Visit**—RefName of the visit, as specified in the XML file that contains the visit definition.
- **Form**—RefName of the CRF, as specified in the XML file that contains the CRF 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 subject 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 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.TextControlRefname*.

**VisitIndex**

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**

Index into a repeating CRF.

- **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**

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 subject 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 clicks 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)

Confirms whether a RefName path is valid, by checking the database for the existence of the specified path. When sites are on different study versions and you want a calculation to update an item that does not exist in all study 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 CRF or an alternate CRF used to capture the same data for subjects who started the CRF in an earlier study 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 CRF rules.

### Arguments

**RefNamePath**

String that identifies the CRF component to get. Each RefName in the string is the RefName specified in the XML file that contains the definition of the CRF component. The REFNAMEPath must be fully qualified; only the subject component of the path defaults to the current subject. The REFNAMEPath argument is in the following format:
(0.Visit.Form.Section.Itemset.Item[.control[.control...]])

- **0**—Current subject.
- **Visit**—RefName of the visit, as specified in the XML file that contains the visit definition.
- **Form**—RefName of the CRF, as specified in the XML file that contains the CRF 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 subject 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 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.TextControlRefname*.

**VisitIndex**

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 CRF, a calculated value is updated on that version; otherwise, the value is updated on an alternate CRF.

```
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, confirms whether a RefName path is valid, by checking the database for the existence of the specified path. When sites are on different study versions and you want a calculation to update an item that does not exist in all study 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 CRF or an alternate CRF used to capture the same data for subjects who started the CRF in an earlier study 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 method, except that it is intended for use in repeating CRFs. Therefore, it has an additional argument, FormIndex, which enables you to reference a specific instance of the repeating CRF.

**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 CRF rules.

### Arguments

**RefNamePath**

String that identifies the CRF component to get. Each RefName in the string is the RefName specified in the XML file that contains the definition of the CRF component. The REFNAMEPath must be fully qualified; only the subject component of the path defaults to the current subject. The REFNAMEPath argument is in the following format:
(0.Visit.Form.Section.Itemset.Item[.control[.control...]])

- **0**—Current subject.
- **Visit**—RefName of the visit, as specified in the XML file that contains the visit definition.
- **Form**—RefName of the CRF, as specified in the XML file that contains the CRF 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 subject 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 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.TextControlRefname*.

**VisitIndex**

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**

Index into a repeating CRF.

- **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 CRF, a calculated value is updated on that version; otherwise, the value is updated on an alternate CRF.

```
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.mtxt
QRSRate",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)

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 CRF component to get. Each RefName in the string is the RefName specified in the XML file that contains the definition of the CRF component. The REFNAMEPath must be fully qualified; only the subject component of the path defaults to the current subject. The REFNAMEPath argument is in the following format:
(0.Visit.Form.Section.Itemset.Item[.control[.control...]])

- **0**—Current subject.
- **Visit**—RefName of the visit, as specified in the XML file that contains the visit definition.
- **Form**—RefName of the CRF, as specified in the XML file that contains the CRF 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 subject 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 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.TextControlRefname*.

**ItemSetIndex**

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 subject 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 clicks 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**

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 CRF, 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 method, except that it is intended for use in repeating CRFs. Therefore, it has an additional argument, FormIndex, which allows you to reference a specific instance of the repeating CRF.

**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 CRF component to get. Each RefName in the string is the RefName specified in the XML file that contains the definition of the CRF component. The REFNAMEPath must be fully qualified; only the subject component of the path defaults to the current subject. The REFNAMEPath argument is in the following format:
(0.Visit.Form.Section.Itemset.Item[.control[.control...]])

- **0**—Current subject.
- **Visit**—RefName of the visit, as specified in the XML file that contains the visit definition.
- **Form**—RefName of the CRF, as specified in the XML file that contains the CRF 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 subject 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 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.TextControlRefname*.

**ItemSetIndex**

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 subject 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 clicks 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**

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**

Index into a repeating CRF.

- **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 CRF 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()

Loads all of the data in the current subject's case book into memory. This provides better performance when a rule must check multiple data items in several visits and CRFs, as the rule can find data without querying the database multiple times.

## RemoveEmptyDynamicVisit

Removes a dynamic visit that a subject has been scheduled into, if no data has been entered into any of the visit's CRFs. The method takes the RefName of the dynamic visit to remove as its only argument.

### Removing a dynamic visit

If no subject data has been entered in any CRF 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 subject'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 subject 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 CRF appears or may appear.
- RefName of the dynamic CRF 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 subject is female. If the gender is not female, the CRF is removed from the visit and will not be scheduled for the subject.

## SetDynamicVisitStart

Schedules a subject 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.
- 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 study.

To assign a subject 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.

### Example 1

In this example, the subject gender determines whether the subject is scheduled for the Dose100 or the Dose250 visit, both of which start 720 hours after the subject'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 subject gender determines whether the subject 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)

Uses the value of the data entered in the specified form component. SetEnteredValue is very similar to the SetValue method, which uses the normalized value of the data entered in the specified CRF component. The difference between these methods applies to CRF components that are defined as units.

The MedML Installer utility allows you to define CRF components that you specify as units. You can define unit CRF 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 CRF 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.

**Note:** The **SYMBOL** attribute of a unit can be translated. When calling the method, the rule or derivation can pass in the value of the unit symbol in any of its translations. It is not necessary to pass in the translation for the CRF locale of the form for which the value is being set.

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 CRF rules.

## Arguments

**RefNamePath**

String that identifies the CRF component to get. Each RefName in the string is the RefName specified in the XML file that contains the definition of the CRF component. The REFNAMEPath must be fully qualified; only the subject component of the path defaults to the current subject. The REFNAMEPath argument is in the following format:
(0.Visit.Form.Section.Itemset.Item[.control[.control...]])

- **0**—Current subject.
- **Visit**—RefName of the visit, as specified in the XML file that contains the visit definition.
- **Form**—RefName of the CRF, as specified in the XML file that contains the CRF 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 subject 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 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.TextControlRefname*.

**AttributeName**

String that specifies the type of information to return, specified in double quotes (""). Valid AttributeNames are:

- **VALUE**—Gets the value of the CRF 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 GetEnteredValue method on the Patient object. **VALIDDATEMAP** returns an integer that contains a bitmask indicating which elements of the DateTime control 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 GetEnteredValue method on the Patient object. **UNKNOWNDATEMAP** returns an integer

that contains a bitmask indicating which elements of the DateTime control 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**

Starting with 0, an index into an array of the selected items in a multiple-value control such as a list of checkboxes. 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**

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 subject 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 clicks 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**

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 subject ID is added to a site number retrieved by SetEnteredValue
v2subno="0.V2.SUBID.SUBID_S.0.SUBNO"
v2scrnno="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 v2scrnno,"",0,0,0,"S"&siteno&pn1
   end if

end if
```

## SetEnteredValueRF(REFNAMEPath,AttributeName, Index, ItemsetIndex, VisitIndex, FormIndex, Value)

In instances of a repeating form, uses the value of the data entered in the specified CRF component. The SetEnteredValueRF method is identical to the SetEnteredValue method, except that it is intended for use in repeating CRFs. Therefore, it has an additional argument, FormIndex, which allows you to reference a specific instance of the repeating CRF.

**Note:** The **SYMBOL** attribute of a unit can be translated. When calling the method, the rule or derivation can pass in the value of the unit symbol in any of its translations. It is not necessary to pass in the translation for the CRF locale of the form for which the value is being set.

Use SetEnteredValueRF only in calculations, not in CRF rules.

## Arguments

**RefNamePath**

String that identifies the CRF component to get. Each RefName in the string is the RefName specified in the XML file that contains the definition of the CRF component. The REFNAMEPath must be fully qualified; only the subject component of the path defaults to the current subject. The REFNAMEPath argument is in the following format:
(0.Visit.Form.Section.Itemset.Item[.control[.control...]])

- **0**—Current subject.
- **Visit**—RefName of the visit, as specified in the XML file that contains the visit definition.
- **Form**—RefName of the CRF, as specified in the XML file that contains the CRF 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 subject 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 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.TextControlRefname*.

**AttributeName**

String that specifies the type of information to return, specified in double quotes (""). Valid AttributeNames are:

- **VALUE**—Gets the value of the CRF 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 GetEnteredValue method on the Patient object. **VALIDDATEMAP** returns an integer that contains a bitmask indicating which elements of the DateTime control 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 GetEnteredValue method on the Patient object. **UNKNOWNDATEMAP** returns an integer that contains a bitmask indicating which elements of the DateTime control 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**

Starting with 0, an index into an array of the selected items in a multiple-value control such as a list of checkboxes. 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**

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 subject 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 clicks 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**

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**

Index into a repeating CRF.

- **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 CRF 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 CRF to other CRFs in the study.

**Note:** The **SYMBOL** attribute of a unit can be translated. When calling the method, the rule or derivation can pass in the value of the unit symbol in any of its translations. It is not necessary to pass in the translation for the CRF locale of the form for which the value is being set.

Do not use SetNewValue in a CRF rule.

### Arguments

**RefNamePath**

String that identifies the CRF component to get. Each RefName in the string is the RefName specified in the XML file that contains the definition of the CRF component. The REFNAMEPath must be fully qualified; only the subject component of the path defaults to the current subject. The REFNAMEPath argument is in the following format:
(0.Visit.Form.Section.Itemset.Item[.control[.control...]])

- **0**—Current subject.
- **Visit**—RefName of the visit, as specified in the XML file that contains the visit definition.
- **Form**—RefName of the CRF, as specified in the XML file that contains the CRF 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 subject 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 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.TextControlRefname*.

**AttributeName**

String that specifies the type of information to use, specified in double quotes (""). Valid AttributeNames are:

- **VALUE**—Gets the value of the CRF 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 GetEnteredValue method on the Patient object. **VALIDDATEMAP** returns an integer that contains a bitmask indicating which elements of the DateTime control 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 GetEnteredValue method on the Patient object. **UNKNOWNDATEMAP** returns an integer that contains a bitmask indicating which elements of the DateTime control are unknown.

| | |
|---|---|
| Year | 1 |
| Month | 2 |
| Day | 4 |
| Hour | 8 |
| Minute | 16 |
| Second | 32 |

**Index**

Starting with 0, an index into an array of the selected items in a multiple-value control such as a list of checkboxes. 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**

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 subject 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 clicks 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**

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 CRF 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 CRF to other CRFs in the study.

The SetNewValueRF method is identical to the SetNewValue method, except that it is intended for use in repeating CRFs. Therefore, it has an additional argument, FormIndex, which allows you to reference a specific instance of the repeating CRF.

**Note:** The **SYMBOL** attribute of a unit can be translated. When calling the method, the rule or derivation can pass in the value of the unit symbol in any of its translations. It is not necessary to pass in the translation for the CRF locale of the form for which the value is being set.

Do not use SetNewValueRF in a CRF rule.

## Arguments

**RefNamePath**

String that identifies the CRF component to get. Each RefName in the string is the RefName specified in the XML file that contains the definition of the CRF component. The REFNAMEPath must be fully qualified; only the subject component of the path defaults to the current subject. The REFNAMEPath argument is in the following format:
(0.Visit.Form.Section.Itemset.Item[.control[.control...]])

- **0**—Current subject.
- **Visit**—RefName of the visit, as specified in the XML file that contains the visit definition.
- **Form**—RefName of the CRF, as specified in the XML file that contains the CRF 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 subject 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 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.TextControlRefname*.

**AttributeName** String that specifies the type of information to use, specified in double quotes (""). Valid AttributeNames are:

- **VALUE**—Gets the value of the CRF 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 GetEnteredValue method on the Patient object. **VALIDDATEMAP** returns an integer that contains a bitmask indicating which elements of the DateTime control 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 GetEnteredValue method on the Patient object. **UNKNOWNDATEMAP** returns an integer that contains a bitmask indicating which elements of the DateTime control are unknown.

| | |
|---|---|
| Year | 1 |
| Month | 2 |
| Day | 4 |
| Hour | 8 |
| Minute | 16 |
| Second | 32 |

**Index**

Starting with 0, an index into an array of the selected items in a multiple-value control such as a list of checkboxes. 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**

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 subject 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 clicks 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**

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**

Index into a repeating CRF.

- **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 CRF to the value of a control in another CRF.

```
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)

Uses the normalized value of the data entered in the specified CRF component. SetValue is very similar to the SetEenteredValue method, which uses the entered value of the data entered in the specified CRF component. The difference between these methods applies to CRF components that are defined as units.

The MedML Installer utility allows you to define CRF components that you specify as units. You can define unit CRF 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 CRF 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.

**Note:** The **SYMBOL** attribute of a unit can be translated. When calling the method, the rule or derivation can pass in the value of the unit symbol in any of its translations. It is not necessary to pass in the translation for the CRF locale of the form for which the value is being set.

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 CRF rules.

### Arguments

**RefNamePath**

String that identifies the CRF component to get. Each RefName in the string is the RefName specified in the XML file that contains the definition of the CRF component. The REFNAMEPath must be fully qualified; only the subject component of the path defaults to the current subject. The REFNAMEPath argument is in the following format:
(0.Visit.Form.Section.Itemset.Item[.control[.control...]])

- **0**—Current subject.
- **Visit**—RefName of the visit, as specified in the XML file that contains the visit definition.
- **Form**—RefName of the CRF, as specified in the XML file that contains the CRF 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 subject 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 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.TextControlRefname*.

**AttributeName**

String that specifies the type of information to use, specified in double quotes (""). Valid AttributeNames are:

- **VALUE**—Gets the value of the CRF 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 GetEnteredValue method on the Patient object. **VALIDDATEMAP** returns an integer that contains a bitmask indicating which elements of the DateTime control 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 GetEnteredValue method on the Patient object. **UNKNOWNDATEMAP** returns an integer that contains a bitmask indicating which elements of the DateTime control 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**

Starting with 0, an index into an array of the selected items in a multiple-value control such as a list of checkboxes. 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**

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 subject 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 clicks 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**

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 CRF, uses the normalized value of the data entered in the specified CRF component. The SetValueRF method is identical to the SetValue method, except that it is intended for use in repeating CRFs. Therefore, it has an additional argument, FormIndex, which allows you to reference a specific instance of the repeating CRF.

**Note:** The **SYMBOL** attribute of a unit can be translated. When calling the method, the rule or derivation can pass in the value of the unit symbol in any of its translations. It is not necessary to pass in the translation for the CRF locale of the form for which the value is being set.

Use SetValueRF only in calculations, not in CRF rules.

## Arguments

**RefNamePath**

String that identifies the CRF component to get. Each RefName in the string is the RefName specified in the XML file that contains the definition of the CRF component. The REFNAMEPath must be fully qualified; only the subject component of the path defaults to the current subject. The REFNAMEPath argument is in the following format:
(0.Visit.Form.Section.Itemset.Item[.control[.control...]])

- **0**—Current subject.
- **Visit**—RefName of the visit, as specified in the XML file that contains the visit definition.
- **Form**—RefName of the CRF, as specified in the XML file that contains the CRF 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 subject 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 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.TextControlRefname*.

**AttributeName**

String that specifies the type of information to use, specified in double quotes (""). Valid AttributeNames are:

- **VALUE**—Gets the value of the CRF 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 GetEnteredValue method on the Patient object. **VALIDDATEMAP** returns an integer that contains a bitmask indicating which elements of the DateTime control 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 GetEnteredValue method on the Patient object. **UNKNOWNDATEMAP** returns an integer that contains a bitmask indicating which elements of the DateTime control 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**

Starting with 0, an index into an array of the selected items in a multiple-value control such as a list of checkboxes. 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**

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 subject 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 clicks 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**

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**

Index into a repeating CRF.

- **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

Inserts a dynamic CRF into a visit based on the outcome of a calculation on another CRF.

ShowDynamicForm has the following parameters:

- RefName of the visit in which to schedule the appearance of the CRF.
- RefName of the CRF 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 subject gender determines whether a pregnancy test CRF 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 3

# InForm Data Import utility

## In this chapter

# Overview of the InForm Data Import utility

## Parts of the import utility

| Part | Description |
|---|---|
| Data file | A pipe-delimited file that contains the following:<br>• Line feed characters and carriage returns between lines.<br>• Data to load into the study database. |
| Map file | A file that contains all the mapping information that is necessary to import the data file to the InForm database. |
| XML file | A file that contains information that can be directly imported into the study database. For more information, see *Creating a data and map file* (on page 339) and *Adding new clinical data* (on page 322). |
| Rules | Information against which the XML file is compared to ensure that it complies with the study standards.<br>You use rules to:<br>• Raise queries about data.<br>• Validate data.<br>**Note:** Running rules might cause the import to run slowly. |
| InForm application server | The server on which the InForm application is running. |
| Study database | The database in which the data for the study 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. To run the import for this input file format, select the **InForm Data and Map files** option in the InForm Data Import main window.

  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 subject records from one site to another. To run the import for this input file format, select the **MedML file** option in the InForm Data Import main window.

  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.

**Note:** The InForm Data Import utility does not support importing Regulatory Documents or Visit Reports data.

## Special considerations

- **Importing new subject data**—You must use the MedML file option to import data for a subject who has not gone through the screening and enrollment process.

  You cannot use the data and map file option to import screening and enrollment data.

  **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 MedML file option.

  To import item-level comments, you must use either the data and map file option or the MedML 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 MedML file or data and map file option.

- **Importing on multiple servers**—If your study 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!dtdatetime!|
    !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 one

time to the form, and the next data entry will go to a new form instance.

- **Importing files that contain non-alphanumeric characters**—All import files that include non-alphanumeric characters, such as Japanese characters, must be in Unicode format. All other formats will be read as ASCII and may yield misleading error messages.
- **Importing data to an Add Entry itemset**—You can add rows to an Add Entry itemset, and you can add or modify data in an existing Add Entry itemset row.
  - To add a row to an Add Entry itemset, use the PATIENTDATA tag. You can add one row at a time to the itemset.

    For more information, see *Adding new patient clinical data* (on page 322).
  - To add or modify data for an item in an Add Entry itemset, use the EDITPATIENTDATA tag.
    - You must use the EDITPATIENTDATA tag to add or modify data for an item in an Add Entry itemset, regardless of whether the item previously contained data.
    - When you use the EDITPATIENTDATA tag, you cannot use the DATA sub-tag with the ITEMSETINDEX tag to modify itemset data. You must use the ITEMSETINDEX attribute with the EDITPATIENTDATA tag.

    For more information, see *Updating existing clinical data* (on page 328).
- **Importing data to a Repeating Data itemset**—You can create the rows in the database for a Repeating Data itemset, and you can add or modify data in an existing Repeating Data itemset row.
  - To create rows in the database for a Repeating Data itemset, use the PATIENTDATA tag. You can create one or more rows at a time for a Repeating Data itemset.

    **Note:** To create rows in a Repeating Data itemset using the PATIENTDATA tag, you must use the ITEMSETINDEX attribute with the DATA sub-tag.

    For more information, see *Adding new patient clinical data* (on page 322).
  - To add or modify data for an item in a Repeating Data itemset, use the EDITPATIENTDATA tag.
    - You must use the EDITPATIENTDATA tag to add or modify data for an item in a Repeating Data itemset, regardless of whether the item previously contained data.
    - When you use the EDITPATIENTDATA tag, you cannot use the DATA sub-tag with the ITEMSETINDEX tag to modify itemset data. You must use the ITEMSETINDEX attribute with the EDITPATIENTDATA tag.

    For more information, see *Updating existing clinical data* (on page 328).
  - Do not use the itemset **!match!** field in a map file, which is used to match an itemset row, when using the **!cd!ItemsetIndex** field. The **!match!** field is generally added manually to a Map file, and does not appear on the Field Definition page of the InForm Data Import utility.
- **Deleting and undeleting Add Entry itemsets**—To delete or undelete Add Entry itemset data, you must use the MedML file option.

  **Note:** You cannot delete or undelete a Repeating Data itemset.

# Importing an XML file

## Overview of importing an XML file

To import an XML file:

1 Create an XML file that contains the subject data to add or update.

For more information, see *Creating an XML import file* (on page 318).

2 Run the import with the MedML file option.

For more information, see *Running the import with the MedML file option* (on page 337).

## Creating an XML import file

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 subject.
- Adding new subject data.
- Updating existing subject data.
- Transferring a subject from one site to another.

To create the import file, use any text editor that creates plain text files. For more information, see *Appendix A: Sample data import XML* (on page 391).

### Creating an XML submission

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 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 subject for whom to import data. For example, to import screening data for a new subject, 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 subject site by mnemonic or by name. If the subject site mnemonic is PF, you would insert the **SITEMNEMONIC** attribute as follows:

```
<SCREEN SITEMNEMONIC="PF">
```

**Note:** If you are using the InForm Data Import utility to transfer subject records between sites, go to the final step. A subject 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 element 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 cannot 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 **"**.

- **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**—The RefName of the selected unit, 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 element.
- **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 element.

For existing subject data:

- **CLEARVALUE**—TRUE, to clear the existing value of the control.

7 Save the file.

## Examples

The following example shows a DATA element that is used to import the initials of subject AAA in the PF site to the Subject initials 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 subject at a site, use the SCREEN element with the following required attribute:

| Attribute | Definition |
|---|---|
| SITEMNEMONIC | Mnemonic of the site at which the subject is being screened. |

### Example

The following sample file illustrates the tags used to screen subject 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 subject at a site, use the ENROLL element with the following required attributes:

| Attribute | Definition |
|---|---|
| PATIENTINITIALS | Initials of the subject that is being enrolled. |
| SITEMNEMONIC | Mnemonic of the site at which the subject is being screened. |
| DUPLICATEORDER | Number that specifies the order in which subjects who have the same subject initials, and were enrolled in the same site were screened. |
| PATIENTNUMBER | Subject number of the subject that is being enrolled. |
| ENROLL | TRUE or FALSE, indicating whether to enroll the subject. |

### Example 1

The following example illustrates the tags used to enroll subject XYZ at site PF. This example assumes that subject 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 subjects with duplicate subject initials should be enrolled. Subjects 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 clinical data

To add new clinical data to an existing subject at a site, use the PATIENTDATA element with the following attributes:

| Attribute | Definition |
|---|---|
| PATIENTINITIALS | Initials of the subject that is being enrolled. Either **PATIENTINITIALS** or **PATIENTNUMBER** is required. |
| PATIENTNUMBER | Subject number of the subject that is being enrolled. Either **PATIENTINITIALS** or **PATIENTNUMBER** is required. |

| Attribute | Definition |
|---|---|
| SITEMNEMONIC | Mnemonic of the site at which the subject is being screened. Either **SITEMNEMONIC** or **SITENAME** is required. |
| SITENAME | Name of the site at which the subject is enrolled. Either **SITEMNEMONIC** or **SITENAME** is required. |
| FORMSETREFNAME | RefName of the visit to which you are importing data. |
| FORMSETINDEX | Indicates to which visit instance to add the data. |
| 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.<br><br>To add an incompletion reason at the form level, do not include a DATA element 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 | Indicates that an association exists. |
| ITEMSETINDEX | Row number for the row to add to an itemset.<br><br>• For an Add Entry itemset:<br>  ▪ Specify one row number at a time.<br>  ▪ Adds a single row to an Add Entry itemset.<br>  ▪ You can use the ITEMSETINDEX attribute with the PATIENTDATA tag or the DATA sub-tag.<br>• For a Repeating Data itemset:<br>  ▪ Specify one or more rows at a time.<br>  ▪ Creates all rows for a Repeating Data itemset.<br>  ▪ For a Repeating Data itemset, you must use the ITEMSETINDEX attribute with the DATA sub-tag.<br><br>**Notes:**<br><br>You cannot specify the ITEMSETINDEX attribute on both the PATIENTDATA and DATA tags.<br><br>ITEMSETINDEX is an optional attribute of the PATIENTDATA tag and the DATA sub-tag. |

### Adding a row to an Add Entry itemset

To add a row to an Add Entry 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 Add Entry itemset row to update. If the ITEMSETINDEX attribute is blank or has a value of 0, the InForm Data Import utility adds a new itemset row. If ITEMSETINDEX is a number other than 0, the InForm Data Import utility updates the specified row. Note that existing itemset data is not changed, but missing data is filled in.<br><br>**Note:** To add or modify data in a row in an Add Entry or a Repeating Data itemset, you must use the EDITPATIENTDATA tag. This is true regardless of whether the row contains data. |

### Creating rows for a Repeating Data itemset

To create rows for a Repeating Data itemset on a form, use the following attributes:

| Attribute | Definition |
|---|---|
| **PATIENTDATA attribute** | **Definition** |
| SECTIONNAME | Specifies the RefName of the section in which the itemset occurs. |
| ITEMSETNAME | Specifies the RefName of the itemset definition. |
| **DATA attribute** | **Definition** |
| ITEMSETINDEX | Specifies one or more itemset row numbers for the rows to create for the Repeating Data itemset.<br><br>Valid values are 1-*n*, where n=the value of the INITIALROWCOUNT of the Repeating Data itemset.<br><br>**Note:** To add or modify data in a row in an Add Entry or a Repeating Data itemset, you must use the EDITPATIENTDATA tag. This is true regardless of whether the row contains data. |

### 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 *Example 3: Adding data to an unscheduled visit* (on page 326). |
| 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 study. |

### 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** attribute.
- To add a comment to a form, include the **COMMENT** attribute along with the text of the comment in a **PATIENTDATA** attribute that does not include **SECTIONNAME** and **ITEMSETNAME** attributes.

### Example: Adding data to a form

The following XML fragment shows elements used to add data to the DEM form for subject 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: Adding a row to an Add Entry itemset

The following XML fragment shows the elements used to create a new row in an Add Entry itemset 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 Add Entry 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.DISCPULLDOWN.DISCPULLDOWN"
      VALUE="Not Effective"/>
</PATIENTDATA>
```

### Example: Creating rows for a Repeating Data itemset

The following XML fragment shows the elements used to create the rows in the Sample Collection itemset on the Dose VS2 form.

The Repeating Data itemset contains two rows with predefined values:

- Hour 10, Minute 0
- Hour 10, Minute 2

The value of 50.000 is entered for the Hour 10, Minute 0 data point.

```
<PATIENTDATA PATIENTINITIALS='RDF' SITEMNEMONIC='01'
FORMSETREFNAME='CommonCRF' FORMREFNAME='frmDOSEVS2' FORMINDEX='0'
SECTIONNAME='sctSampleCollection' ITEMSETNAME='sctSampleCollection'>
   <DATA TAG='sctSampleCollection.sctSampleCollection.itmTime.itmTime'
   ITEMSETINDEX='1' HOUR='10' MINUTE='0' />
   <DATA
   TAG='sctSampleCollection.sctSampleCollection.itmMeasurement.itmMeasurement
   ' ITEMSETINDEX='1' VALUE='50.00' />
   <DATA TAG='sctSampleCollection.sctSampleCollection.itmTime.itmTime'
   ITEMSETINDEX='2' HOUR='10' MINUTE='2' />
</PATIENTDATA>
```

### Example: 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: 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: 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 clinical data

To modify existing clinical data for a subject at a site, use the EDITPATIENTDATA element with the following attributes:

| Attribute | Definition |
|---|---|
| PATIENTINITIALS | Initials of the subject that is being enrolled. Either **PATIENTINITIALS** or **PATIENTNUMBER** is required. |
| PATIENTNUMBER | Subject number of the subject that is being enrolled. Either **PATIENTINITIALS** or **PATIENTNUMBER** is required. |
| SITEMNEMONIC | Mnemonic of the site at which the subject is being screened. Either **SITEMNEMONIC** or **SITENAME** is required. |
| SITENAME | Name of the site at which the subject is enrolled. Either **SITEMNEMONIC** or **SITENAME** is required. |
| FORMSETREFNAME | 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 that all following <Data Tags> are ignored. |
| FORMINDEX | Required for repeating forms.<br>Indicates a repeating form to update. Value corresponds to the form instance to update. |
| ASSOCIATION | Indicates that an association exists. |

| Attribute | Definition |
|---|---|
| ACTION | ADD or REMOVE; specifies whether to add or remove an association. |
| ITEMSETINDEX | Row number for the item in an Add Entry or Repeating Data itemset to modify or add data to.<br><br>**Note:** You cannot use the ITEMSETINDEX with the DATA sub-tag of the EDITPATIENTDATA tag. |

### Editing data in an Add Entry itemset

To edit data in an Add Entry 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 Add Entry itemset row to update. If the ITEMSETINDEX attribute is blank or has a value of 0, the InForm Data Import utility adds a new itemset row. If ITEMSETINDEX is a number other than 0, the InForm Data Import utility updates the specified row. Note that existing itemset data is not changed, but missing data is filled in.<br><br>**Note:** To add or modify data in a row in an Add Entry or a Repeating Data itemset, you must use the EDITPATIENTDATA tag. This is true regardless of whether the row contains data. |

### Editing data in a Repeating Data itemset

To edit data in a Repeating Data 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 | Adds or modifies data for an item in a Repeating Data itemset.<br><br>Specifies the row number for the item in a Repeating Data itemset to modify or add data to.<br><br>Valid values are 1-*n*, where n=the value of the INITIALROWCOUNT of the Repeating Data itemset.<br><br>**Note:** To add or modify data in a row in an Add Entry or a Repeating Data itemset, you must use the EDITPATIENTDATA tag. This is true regardless of whether the row contains data. |

### 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: 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:** 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: 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: Modifying an Add Entry itemset row

The following XML file fragment modifies an Add Entry itemset on the Adverse Events (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>
```

### Example: Modifying a Repeating Data itemset

The following XML file fragment modifies a Repeating Data itemset on the Dose VS2 form.

```
   <EDITPATIENTDATA PATIENTINITIALS='RDF' SITEMNEMONIC='01'
   FORMSETREFNAME='CommonCRF' FORMREFNAME='frmDOSEVS2' FORMINDEX='1'
   SECTIONNAME='sctSampleCollection' ITEMSETNAME='sctSampleCollection'
   ITEMREFNAME='itmTime' ITEMSETINDEX='1' REASONPULLDOWN='1'>
      <DATA TAG='sctSampleCollection.sctSampleCollection.itmTime.itmTime'
      HOUR='11' MINUTE='11' />
   </EDITPATIENTDATA>
   <EDITPATIENTDATA PATIENTINITIALS='RDF' SITEMNEMONIC='01'
   FORMSETREFNAME='CommonCRF' FORMREFNAME='frmDOSEVS2' FORMINDEX='1'
   SECTIONNAME='sctSampleCollection' ITEMSETNAME='sctSampleCollection'
   ITEMREFNAME='itmTime' ITEMSETINDEX='2' REASONPULLDOWN='1'>
      <DATA TAG='sctSampleCollection.sctSampleCollection.itmTime.itmTime'
      HOUR='11' MINUTE='13' />
   </EDITPATIENTDATA>
```

### Example: 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: 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: Deleting an association

The following XML file fragment shows a 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: 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: 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 subject record

You can use the InForm user interface to transfer subject information one subject at a time. You can also transfer subjects in bulk using the InForm Data Import utility.

You can transfer subjects who:

- Change permanent address before completing the study.
- Have multiple residences throughout the course of the study.
- Were initially assigned to the wrong sites or to an investigator who is no longer with the study.

Whether you transfer subjects individually, or transfer them in bulk, keep in mind that:

- The InForm application only allows you to transfer subjects from one site to another if the study version at the destination site is the same or greater than the study version at the current subject site.

  You can view the study version for a site in the Admin interface in the InForm user interface.

  You can transfer only subjects who are fully enrolled; you cannot transfer a subject who is screened but not enrolled or a subject who has failed enrollment.

To transfer a subject record from one site to another, use the following elements:

- PATIENTSITECHANGE
- NEWSITE
- CURRENTSITE

### PATIENTSITECHANGE

The **PATIENTSITECHANGE** element identifies each subject to transfer. Use one pair of opening and closing **PATIENTSITECHANGE** elements for each subject to transfer. The **PATIENTSITECHANGE** element surrounds all information about a single subject, 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 subject transfer. |

Within the **PATIENTSITECHANGE** element, include these elements:

- NEWSITE
- CURRENTSITE

### NEWSITE

The **NEWSITE** element provides information about the destination site for the subject. 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** element has one optional attribute:

| Attribute | Definition |
|---|---|
| PATIENTNUMBER | The subject number for the subject at the destination site. Use this attribute only if you need to change the subject 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 subject. 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 subject 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 subject to be transferred:

| Attribute | Definition |
|---|---|
| PATIENTINITIALS | The initials of the subject being transferred. If you use this attribute, and you know that more than one subject at the current site has the specified initials, use the optional **DUPLICATEORDER** attribute to indicate which subject to transfer. |
| PATIENTNUMBER | The subject number of the subject being transferred, as it exists on the current site. |

Use the following attribute with the **CURRENTSITE** element if more than one subject at the site has the initials specified in the **PATIENTINITIALS** element:

| Attribute | Definition |
|---|---|
| DUPLICATEORDER | Specifies which subject should be transferred, if more than one subject at the site has the same initials. When multiple subjects have the same initials, the InForm application checks the screening numbers of those subjects and transfers the subject whose screening number is in the order specified by the **DUPLICATEORDER** tag.<br><br>Use this attribute only if you have used **PATIENTINITIALS** to identify the subject. |

## Example

This example shows the elements in a MedML file that is used to transfer several subjects. Note that each pair of **PATIENTSITECHANGE** attributes defines current and destination site information for one subject.

```
<?xml version="1.0"?>
<CLINICALDATA>
    <PATIENTSITECHANGE REASON="new patient address">
        <NEWSITE SITEMNEMONIC="MCLEAN"/>
        <CURRENTSITE SITEMNEMONIC="PF" PATIENTNUMBER="1003"/>
    </PATIENTSITECHANGE>
```

<!----For subject1001 the file specifies a new subject number because a subject already exists at the McLean Hospital site with the same subject number.>

```
    <PATIENTSITECHANGE REASON="new patient address">
        <NEWSITE SITENAME="McLean Hospital" PATIENTNUMBER="1002"/>
        <CURRENTSITE SITEMNEMONIC="PF" PATIENTNUMBER="1001"/>
    </PATIENTSITECHANGE>
```

<!----The **DUPLICATEORDER** attribute indicates that subject DDD is the second subject with those initials to be screened at the McLean Hospital site.>

```
    <PATIENTSITECHANGE REASON="new patient address">
        <NEWSITE SITEMNEMONIC="MCLEAN"/>
        <CURRENTSITE SITENAME="McLean Hospital" PATIENTINITIALS="DDD"
            DUPLICATEORDER="2"/>
    </PATIENTSITECHANGE>
</CLINICALDATA>
```

## Additional InForm Data Import utility attributes

| Attribute | Description | Values |
|---|---|---|
| CLEARCRF | Indicates whether to clear the specified form. | • TRUE<br>• FALSE |
| COMMONFORM | Indicates whether a form is a common CRF. | • TRUE<br>• FALSE |
| DELETEITEMSET | Indicates whether to delete an Add Entry itemset. | • TRUE<br>• FALSE |
| FORMSTATUS | Modifies the status of a form. | • FREEZE<br>• UNFREEZE<br>• LOCK<br>• UNLOCK<br>• DELETE<br>• UNDELETE |
| OVERRIDE | Specifies whether a subject screening or enrollment was overridden. | • TRUE<br>• FALSE |
| UNDELETEITEMSET | Indicates whether to undelete a deleted Add Entry itemset. | • TRUE<br>• FALSE |

## Running the import with the MedML file option

**Note:** To import data into the database, the InForm server must be running before you start the InForm Data Import utility.

To import an XML file into the database and submit it through the InForm application server:

1 Click **Start** > **All Programs** > **Oracle® Health Sciences** > **InForm 6.0** > **InForm Data Import**.

The InForm Data Import main window appears.

2 Select **MedML file**.

3 In the **Trial Name** field, enter the name of the study into which to import the file. The last 10 studies you accessed appear 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 Optionally, to run the import without actually importing data into the database, select **Parse Only**. Use this function to test the syntax of your MedML file before you import it into the database.

7 Click **Next**.

A dialog box appears and requests your InForm name and password.

**Note:** If you selected Parse Only, you do not need to specify a 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 subject clinical data | Enter Data into a CRF |
| Update subject clinical data | Edit Data on a CRF |

9 In the **Password** field, type the user password, and click **Next**.

The Summary window appears.

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.

# Importing a data and map file

## Specifying a data and map file

To import a data and map file:

1 Do one of the following:

- Create a map file to import.

  For more information, see *Creating a data and map file* (on page 339).

- Edit an existing map file to import.

  For more information, see *Editing an existing data and map file* (on page 341).

2 Run the import with the InForm Date and Map files option.

  For more information, see *Running the import with the InForm Data and Map files option* (on page 349).

### Creating a data and map file

The data and map file that are used in the direct import method must be a text file. You can use any text editor that creates plain text files, or develop a conversion utility 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 subject number and initials in the following format:

    **subject_number (subject_initials)**

    If it is possible that more than one subject could have the same subject number and initials, you must also include a field for the site mnemonic of the subject.
  - The database ID of the subject.

- 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 340).
- *Time fields* (on page 340).
- *DateTime fields* (on page 340).
- *Nested Controls* (on page 341).
- *Checkboxes and multiple-selection drop-down lists* (on page 341).
- *Units* (on page 341).
- *Item comments* (on page 341).

### 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 must consist of three fields: month, day, and year. The year must use four digits, and times must use a 24-hour clock. Observe these formatting considerations:

- By default, the date order format must match the order specified in the study configuration. If data is entered in a date order format that is different from the study configuration setting, you can override the setting by using the -datetimeformat command line parameter during import. For more information, see *Running the InForm Data Import utility from the command line* (on page 353).
- 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
```

## Editing an existing data and map file

To edit an existing import map file:

**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.

1 Click **Start** > **All Programs** > **Oracle® Health Sciences** > **InForm 6.0** > **InForm Data Import**.

The InForm Data Import main window appears.

2 Select **InForm Data and Map files**.

3 In the **Trial Name** field, enter the name of the study into which to import the file. The last 10 studies you accessed appear in the drop-down list.

4 Click **Next**.

A dialog box appears where you specify the data and map files to import.

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**.

**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 as the pfimport.exe file.

7 Click **Edit Map File**.

The Field Definition dialog box appears.

8 To edit the map file:

a Specify whether the data is targeted for CRF items or CRF itemsets. For more information, see *Specifying a submission type* (on page 342).

b Specify the input field type. For more information, see *Specifying an input field type* (on page 343).

c Specify the definition of each map file, one field at a time. For more information, see *Building an item path* (on page 344).

d Specify the data type. For more information, see *Specifying a data type* (on page 346).

e 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 346).

f Indicate whether the data contains multiple selection items. For more information, see *Indicating data contains multiple selection items* (on page 347).

g Check for duplicate information within itemsets. For more information, see *Checking for duplicate information within itemsets* (on page 347).

h Import information to an unscheduled visit. For more information, see *Importing information into an unscheduled visit* (on page 348).

i Insert or delete an import field into the import file. For more information, see *Inserting or deleting an import field* (on page 348).

j Check the map against the import file. For more information, see *Checking the map against the import file* (on page 348).

9 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 349).

### Specifying 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 import data that only updates existing unscheduled visits, select **Disallow New Repeating Visit Instances**.

3 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.

4 Click **Continue**.

The Field Definition dialog box appears.

5 To define map fields, follow the procedure in *Specifying an input field type* (on page 343).

### Specifying an input field type

Use the Field Definition dialog box to define each map field one at a time. Each map field corresponds to a data item in the import file.

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 344).
- **Patient Field – Number (Initials)**—Contains subject identification information in either patient_number (patient_initials) format or in the form of a subject database ID. If your import data identifies each subject by the subject 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 349).
- **Site Mnemonic Field**—Contains the mnemonic of the site where the subject 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 349).
- **Itemset Index Field**—Contains the itemset index, a 1 based row number. Used to identify an existing row of an Add Entry or Repeating Data 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 349).
- **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 349).
- **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 349).

- **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 349).

Optionally, select one or more of the following checkboxes:

- **Comma separates multiple-values**—Indicates that the import file contains data which will populate multiple-selection controls such as radio button groups, checkbox groups, and drop-down lists.
- **Match Itemset instance with this field**—Indicates that you want the InForm Data Import utility to check for duplicate data in the import file before importing to the InForm application.
- **Match Repeating visit instance with this field**—Indicates that you want the InForm Data Import utility to check for duplicate dates of visit for the current field in the import file and a date of visit for a repeating visit already in the InForm database.

## 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, do one of the following:

- Type the path in the **InForm Item Path** text box.

  For more information, see *Entering an item path explicitly* (on page 344).

- Select each RefName from the drop-down lists in the Build Path From Database dialog box.

  For more information, see *Using the Build Path from Database dialog box* (on page 345).

### 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 subject.
- **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

- The item path for the TEMPTEXT field in the TEMPTEXT item in the VS section of the VSL form in a visit called VISIT1 appears as follows:

```
0.Visit1.VSL.VS.0.TEMPTEXT.TEMPTEXT
```

- The item path for an item in an itemset appears as follows:

```
0.Visit1.SS.SECTION2.SS2GROUP.ONSETDATE.ONSETDATE
```

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.

- The item path for a text control nested within the VIEW radio control item in the CHESTXRAY section of the ECG form in VISIT1 appears as follows:

```
0.Visit1.ECG.CHESTXRAY.0.VIEW.VIEWRADIO.OTHERTEXT
```

The item is identified by the VIEW RefName; the radio control is identified by VIEWRADIO, and the text control is identified by OTHERTEXT.

### 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**.

## 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** section.

  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 InForm Data Import 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**.

## 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 box, select **Comma separates multiple-values**.

## 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.

For example, if you are importing lab information and the subject 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.

To use this feature:

1 In the Submission Type window, select **Itemset**.

2 Click **Continue**.

The Field Definition dialog box appears.

3 Click **Next** to navigate to the data object.

4 Select **InForm Item Path** for the data object to match to an itemset in the database.

5 Select **Match Itemset instance with this field**.

6 Repeat these steps for any other data items you want to compare.

### Importing 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 to match to a repeating visit in the database.

4 Select **Match Repeating visit instance with this field**.

5 Repeat these steps for any other unscheduled visits to import.

### Inserting or deleting an import field

1 In the Field Definition dialog box, click **Insert Field**.

The InForm Data Import utility clears the data entry fields on the Field Definition dialog.

2 Type the definition for the new field.

3 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 Field**.

4 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 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 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:

- In the Field Definition dialog, navigate through the field definitions and compare the data that appears in the **Sample Data** field with the data for the import file.

### 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**.

## Running the import with the InForm Data and Map files option

**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 Click **Start** > **All Programs** > **Oracle® Health Sciences** > **InForm 6.0** > **InForm Data Import**.

The InForm Data Import main window appears.

2 Select **InForm Data and Map files**.

3 In the **Trial Name** field, enter the name of the study into which to import the file. The last 10 studies you accessed appear in the drop-down list.

4 Click **Next**.

A dialog box appears where you specify the data and map files to import.

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 Optionally, to edit the map file, click **Edit Map File**. For more information, see *Editing an existing data and map file* (on page 341).

8 Click **Next**.

A dialog box appears and requests your InForm name and password.

**Note:** If you selected **Parse Only**, you do not need to specify a 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 subject clinical data | Enter Data into a CRF |

| To import data that matches this InForm system activity | User needs these rights |
|---|---|
| Update subject clinical data | Edit Data on a CRF |

10 In the **Password** field, type the user password, and 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.

# Checking 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, PFImport.exe, is stored. After the import is complete, check for the presence of an ERR file and review the file for errors.

# 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:<br>`ss:mm:hh dd/mm/yy`<br>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. |

# 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, you can run the utility in the PFConsole utility, from the command line, or you can insert the import command in a batch file that is scheduled to run periodically.

**Note:** Oracle recommends that you run the InForm Data Import utility through the PFConsole utility. For more information, see ***Running the PFConsole utility from the command line*** (on page 3).

Use the following command line parameters. You must include a space between the parameter name and its value.

```
PFImport [-?] [-autorun] [-verbose] [-parse] [-trial <trialname>] [-errstop]
-norules
[[/accountparams:path_to_password_file]
 [@<rsp_file>] [-xml
<xml_file>...<xml_file>]] [-template <map_file>] [-import <data_file>]
```

**Note:** All command line applications, such as the MedML Installer utility, the InForm Data Import utility, and the InForm Data Export utility use the default product locale specified during the InForm installation, or through PFAdmin commands.

| 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 | | The InForm Data Import utility will write detailed messages as it processes the input file. |
| -parse | | 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 | *trialname* | The study into which you are importing data. Use the full pathname of the study. |
| -errstop | | 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 | | Rules will not run during the import. This parameter is required. This option is valid only if there are no synchronization connections defined. |

| Parameter | Variable | Description |
|---|---|---|
| /accountparams:*path_to_password_file* | | When specified, includes the path to a text file that contains the user name and passwords required to run the command.<br><br>If the accountparams option is not specified, the command prompts for the required user names and passwords.<br><br>The format of the parameter file is parameter=value. There is a new line for each parameter, and there are no spaces on a line.<br><br>**Note:** You must manually create the text file. |
| -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. |
| @ | *RSP_file* | 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 | *xml_file* | 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. |
| -template | *map_file* | 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 | *data_file* | 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. |
| -datetimeformat | | Format to use for date and time information. Use this parameter to override the study configuration settings for date order format. Valid options are:<br><br>• MMDDYYYY<br>• DDMMYYYY<br>• YYYYMMDD |
| -output | output_file | 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 out of the InForm application server**—Log out of the InForm application server before running the import. You can keep the server running.
- **Stop the WWW Publishing Service**—Stop and restart the WWW Publishing Service to prevent others from entering data as you perform an import.
- **Run the Oracle update statistics script**—Run the Oracle update statistics script immediately after you import files.
- **Change the Home page**—Before you start the import, change the Home page of the study and warn users that they may experience slowness in the system while you are running the import.
- **Organize by subject**—If possible, sort the data you are importing by subject ID.

CHAPTER 4

# InForm Data Export utility

## In this chapter

# Overview of the InForm Data Export utility

## InForm Data Export utility output options

- **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.

**Note:** Use of the InForm Data Export utility is limited to studies with fewer than 100,000 subjects.

You can use the CRF Submit application to export data in PDF format. The CRF Submit application is a stand-alone utility that allows you to:

- Create PDFs of subject data for FDA submission purposes. For example, you can create PDFs of subject data to submit to the FDA along with a New Drug Application (NDA).
- Archive PDFs of clinical data internally. For example, you can archive all clinical data for a single subject prior to moving the subject to a new site.

# Running the InForm Data Export utility

All 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 Sciences** > **InForm 6.0** > **InForm Data Export**.

The InForm Data Export main window appears.

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, select a format to use for the export:

- CDD
- Name Value Pairs

For more information, see *InForm Data Export utility output options* (on page 358).

4 In the **Trial Name** field, type the name of the study from which 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 data into a CDD* (on page 360).
- *Exporting Name Value* (on page 364).

# Exporting data into a CDD

## Overview of exporting data into a CDD

The CDD output option reads the CDD mapping parameters and all the data in the InForm database, then populates the CDD in the following order:

- Site data
- Comments
- Subject information
- CRF data

This is useful for repopulating the database when the CDD definition changes after a study starts.

## Moving data to a CDD

1 Use the Central Designer application to generate a CDD definition that includes parameters that specify how to map data from the InForm database to the CDD.

2 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 360).

## 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.**

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 study 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 In the **DSN** field, do one of the following:

- Enter the name of the ODBC DSN that is defined for the CDD to which to output data.
- Create a new DSN.

For more information, see *Creating a new DSN for the export* (on page 361).

7 In the **User Name** and **Password** fields, type the user name and password that is used to access the CDD.

8 To drop the current database and create a new schema based on the RefName associated with the DSN, select the **New Schema** checkbox.

9 In the **Table Space** field, type the name of the tablespace in which to put the new schema.

10 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.**

11 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

1 In the CDD Export Options window, type the following information:

- In the **DSN** field, type the name for the ODBC DSN to create.
- In the **User Name** and **Password** fields, type the user name and password for accessing the CDD.

2 Click **Create DSN**.

The CDD Data Source window appears. Your DSN appears in the **CDD DSN** field.

3 From the **CDD RefName** drop-down list, select the RefName to associate with the new DSN.

4 In the **Database Server** field, type the name of the server to use.

5 Click **OK**.

The CDD Export Options window reappears, and the DSN, user name, and password appear in the appropriate fields.

**Note: If you create a new DSN for the export, the new DSN is valid only in the InForm Data Import 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 tool, use the ODBC Manager or PFAdmin CONFIG CDD Setup commands. For more information on PFAdmin commands, see the *Study and Reporting Setup Guide*.**

# Exporting name value pairs

## Overview of 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 subject data; it does not export values from the Reg Docs or Visit Reports forms.

## 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 |
|---|---|
| Subject identifier | Subject initials followed by subject number in parentheses. |
| RefName path | Path of RefNames in the following order:<br>• Visit.<br>• 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.<br>• Form.<br>• Repeating form index.<br>• Section.<br>• Itemset.<br>• Itemset index, which is used to indicate which row of an itemset is referenced.<br>• Item.<br>• Control. |
| 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 |
|---|---|
| Subject identifier | Subject initials followed by subject number in parentheses. |
| RefName path | Path of RefNames in the following order:<br>• Visit.<br>• Repeating formset index, which is used to indicate which instance of an unscheduled visit is referenced. If the visit is not an unscheduled visit, the value is 1.000.<br>• Form.<br>• Repeating form index.<br>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. |
| 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 subject 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|CMHidde
nJ.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"
```

## Running the export for name value pairs

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 study 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:

- To instruct the InForm Data Export utility not to output data from deleted forms or itemset rows, select **Do not output data from deleted Forms or Itemset rows**.
- To export data from deleted forms or itemset rows:
  1 Select **Output Deleted Forms and Itemset Rows**.
  2 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.

# 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 recommends that you run the InForm Data Export utility through PFConsole utility. For more information, see ***Running the PFConsole utility from the command line*** (on page 3).

**Note:** All command line applications, such as the MedML Installer utility, the InForm Data Import utility, and the InForm Data Export utility use the default product locale specified during the InForm installation, or through PFAdmin commands.

| Parameter | Variable | Description |
|---|---|---|
| PFConsole utility | | Starts PFConsole utility. |
| PFExport | | Starts the InForm Data Export utility. |
| /accountparams:*path_to_password_file* | | When specified, includes the path to a text file that contains the user name and passwords required to run the command.<br><br>If the accountparams option is not specified, the command prompts for the required user names and passwords.<br><br>The format of the parameter file is parameter=value. There is a new line for each parameter, and there are no spaces on a line.<br><br>**Note:** You must manually create the text file. |
| -autorun | | Runs the InForm Data Export utility in a command window.<br><br>For more information, see *Running the PFConsole utility from the command line* (on page 3). |
| -? | | Displays the usage statement. |
| -help | | Displays the online Help for the InForm Data Export utility. |
| -Trial | trial name | The name of the study from which to export data. |
| -outfile | output log file name | Indicates that the next parameter is the name of an output file.<br><br>Full pathname of an output file that contains the text of messages displayed by the InForm Data Export utility. Optional. |

| Parameter | Variable | Description |
|---|---|---|
| -cdd | ODBC DSN Name | Indicates that you are exporting data into a CDD.<br>ODBC DSN name of the CDD to use as the export target of the CDD Output tool is required. |
| -CreatDSN | OraConnStr | For CDD export only, creates a new DSN within the CDD database. |
| -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.<br>Name of the export file that is created when you run the export tool 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. |
| -OutputUN | | Outputs UN for unknown date fields, if present. |
| -sysadmin | | Name of the System Administrator with rights to export CRF Help.<br>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. |

## 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]  [/accountparams:path_to_password_file]

[-CreatDSN OraConnStr][-RefName CDDRefName][-TBSP OraTabSpace]
[-crfhelp]
[-nvfile nv file name [-DelPrefix string to prefix data marked as deleted]]

[-OutputUN]]
```

## DEL file format

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 subjects.

```
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 5

# InForm Performance Monitor utility

## In this chapter

# Overview of the InForm Performance Monitor utility

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 study 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 to 10 minute intervals.

# Starting the InForm Performance Monitor utility

- Click **Start** > **Programs** > **Oracle® Health Sciences** > **InForm 6.0** > **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.

**Note:** The InForm Performance Monitor utility is used by Oracle to assess and enhance the performance of a study, 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 372), the use and interpretation of the InForm Performance Monitor utility is reserved by Oracle.

# 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 study that is associated with an InForm Performance Monitor utility message. If a message is not study-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.
- **Time (ms)**—Elapsed time in milliseconds from the user-issued request to the server response.
- **Time of transaction**—The time at which the message was posted, in Greenwich Mean Time (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

1 Select **View** > **Options**.

The Performance Monitor Options window appears.

2 In the **Messages to View** section, select one or more of the following filters to receive messages from only the specified subsystems.

- To select multiple subsystems, hold down the **Ctrl** key while you select.
- To select all options, click **Select All**.
- To clear the list of options, click **Clear All**.

| Filter | Subsystem |
|---|---|
| IIS Request Data | The InForm ISAPI. |
| Trial package | The InForm study MTS package, by the InForm application server. |
| Rule package | The InForm rule package, by the InForm application server. |
| CDD package | The InForm CDD package, by the InForm application server. |
| UDA | The InForm database. |
| ODBC | The InForm ODBC. |
| Synchronization | Not supported. |
| Reports | Not supported. |
| Listing | The InForm listing. |
| InForm service | The InForm service. |
| PFImport | The InForm Data Import utility. |

3 In the **Minimum Time** section, type the minimum number of milliseconds an action should take in

order to be included in the list of messages.

4 Click **OK**.

## Viewing messages from specific InForm servers

1 Select **View** > **Servers**.

The InForm Server Selection List dialog box appears.

2 Select the server from which to capture messages.

- To select multiple servers, hold down the **Ctrl** key while you select.
- To select all servers, click **Select All**.
- To clear the list of servers, click **Clear All**.

3 Click **OK**.

# InForm Performance Monitor utility output options

**Note:** Before you select an output option, select the statistics to capture as described in ***Capturing performance statistics*** (on page 372).

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 **Output** > **Output To Window** |
| Stream output to a file | Sends messages to a file instead of (or in addition to) displaying them in the Performance Monitor window. | Select **Output** > **Stream To File** |
| Save a performance log | Saves the performance log as comma-separated text to a specified file. | Select **File** > **Save As** |

# Managing the InForm Performance Monitor utility data

| Task | Action |
|---|---|
| Sort a column | Click the column heading bar. |
| Select a single message | Click the message. |
| Select multiple contiguous messages | Hold down the **Shift** key while clicking the messages. |
| Select multiple noncontiguous messages | Hold down the **Ctrl** key while clicking the messages. |
| Select all messages | Select **Edit** > **Select All**. |
| Copy selected messages | Select **Edit** > **Copy**, or click the **Copy** button in the toolbar. |
| Delete selected messages | Select **Edit** > **Delete**, or click the **Delete** button in the toolbar. |
| Clear all messages | Select **Edit** > **Reset**. |

# Examples of using the InForm Performance Monitor utility

## Testing rule script efficiency

After you develop a set of rules for a study and add subject 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

1 In the Performance Monitor window, select **View** > **Options**.

The Performance Monitor Options dialog box appears.

2 Clear all filters except **Rule Package** to set the message filter to capture rule data.

3 Click **OK**.

4 Start the InForm application for your study.

5 Navigate to an item that is associated with a rule.

6 Edit the item, but do not click **Submit**.

7 In the Performance Monitor window, click **Edit > Reset** to clear the display.

8 In the InForm application, click **Submit**.

9 Return to the Performance Monitor window and check the messages.

Messages appear, indicating that:

- The 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 as your study 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 to 10 minute intervals.

## Capturing SQL query performance data

1 From the Performance Monitor window, select **View** > **Options**.

The Performance Monitor Options dialog box appears.

2 Clear all filters except **UDA** to set the message filter to capture rule data.

3 Click **OK**.

4 Start the InForm application.

5 Navigate to an item that is associated with a query.

6 Edit the item, but do not click **Submit**.

7 In the Performance Monitor window, select **Edit > Reset** to clear the display.

8 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.

9 In the InForm application, click **Submit**.

10 Return to the Performance Monitor window and check the messages.

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, and compare the data in the Time field on the Performance Monitor window for each captured message.

CHAPTER 6

# InForm Report Folder Maintenance utility

## In this chapter

# Overview of the InForm Report Folder Maintenance utility

The InForm Report Folder Maintenance utility is a Windows application that can do the following:

- Copy Reporting and Analysis folders and their contents to target folders.
- Define and create folder structures when using a single Reporting and Analysis server for any of the following:
  - Multiple studies.
  - Multiple sponsors.
  - Multiple studies within multiple sponsors.
- Update any links to drill-down reports in InForm standard reports that were copied to a target location.
- Associate a copied report with a new Reporting and Analysis package at the target location.

  **Note:** This includes saved reports as well as standard report definitions.

You must have the Modify System Configuration right to run the InForm Report Folder Maintenance utility.

You install the InForm Report Folder Maintenance utility on the Reporting and Analysis (Cognos 8 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 the InForm Trial Management package, or any published study-specific clinical package. For more information about the association between reports and report packages, see *Report package association* (on page 387).

**Note:** You must configure a study in the Reporting and Analysis before performing the procedures in this chapter. For more information, see the *Study and Reporting Setup Guide*.

# Folder structure for multiple studies or sponsors

## Setting up the initial folder structure

1 Import the standard reports archive. This archive includes all standard reports, as well as the InForm Trial Management package.

2 Publish a study-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, and the reporting packages. These folders appear in the **Public Folders** tab of the Reporting and Analysis portal. The following folders appear after you initially set up a single study:

- CRF Reports
- Item Reports
- Query Reports
- Subject Reports
- Audit Trail Reports
- InForm Trial Management packages
- <*study_name*> clinical package

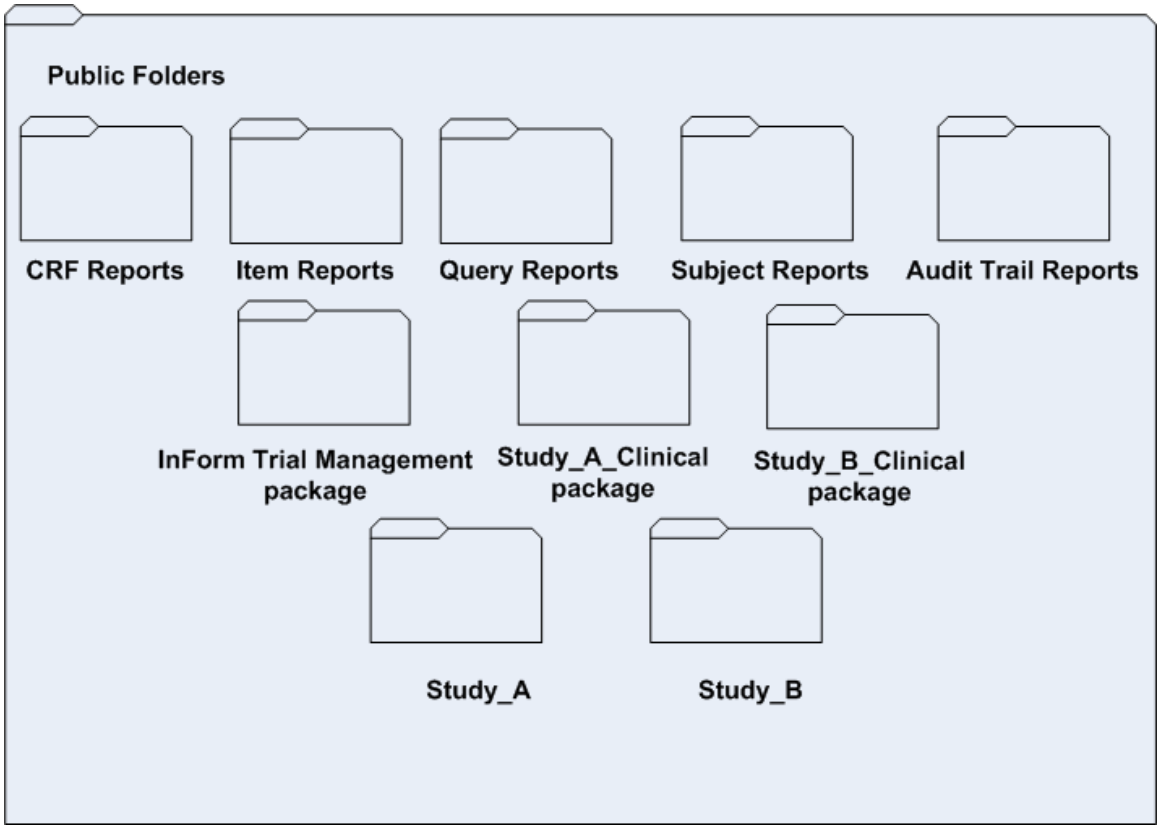
## Folder structure for multiple studies

To set up several studies for a single sponsor, consider creating a separate folder for each study. PFRInit, the utility that you run when you set up the Reporting and Analysis for a study, does this automatically.

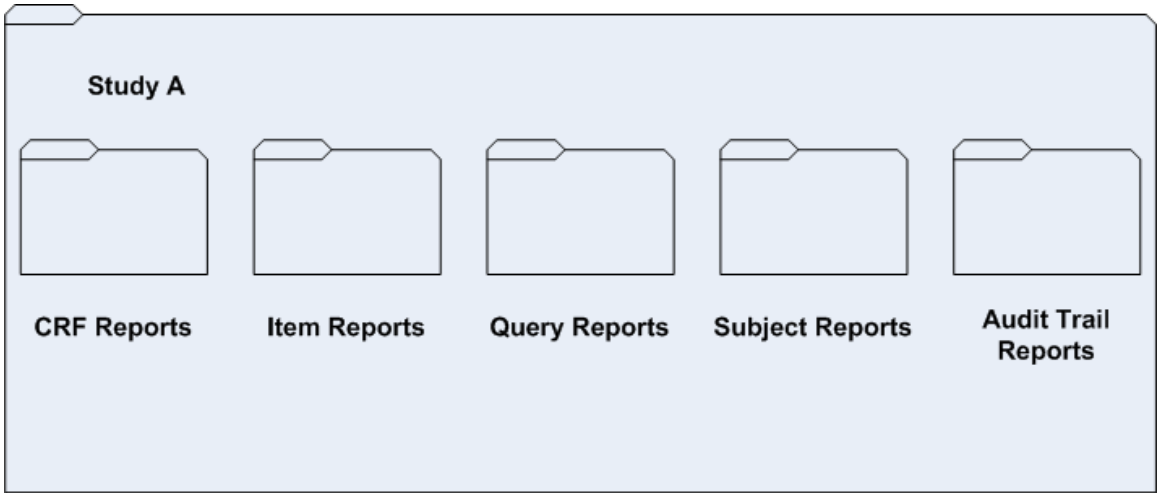
The following shows what the **Public Folders** portal might contain in the Reporting and Analysis portal with folders for two studies (Study A and Study B).

Note that this configuration uses three different packages:

- **InForm Trial Management package**—Shared by both Study_A and Study_B.
- **Study__A_Clinical package**—Used only for Study_A.
- **Study__B_Clinical package**—Used only for Study_B.



The contents of each study-specific folder are set up to include the default reporting folder structure.




This structure ensures that you can save study-specific properties, such as prompt values and report schedules, for a specified report.

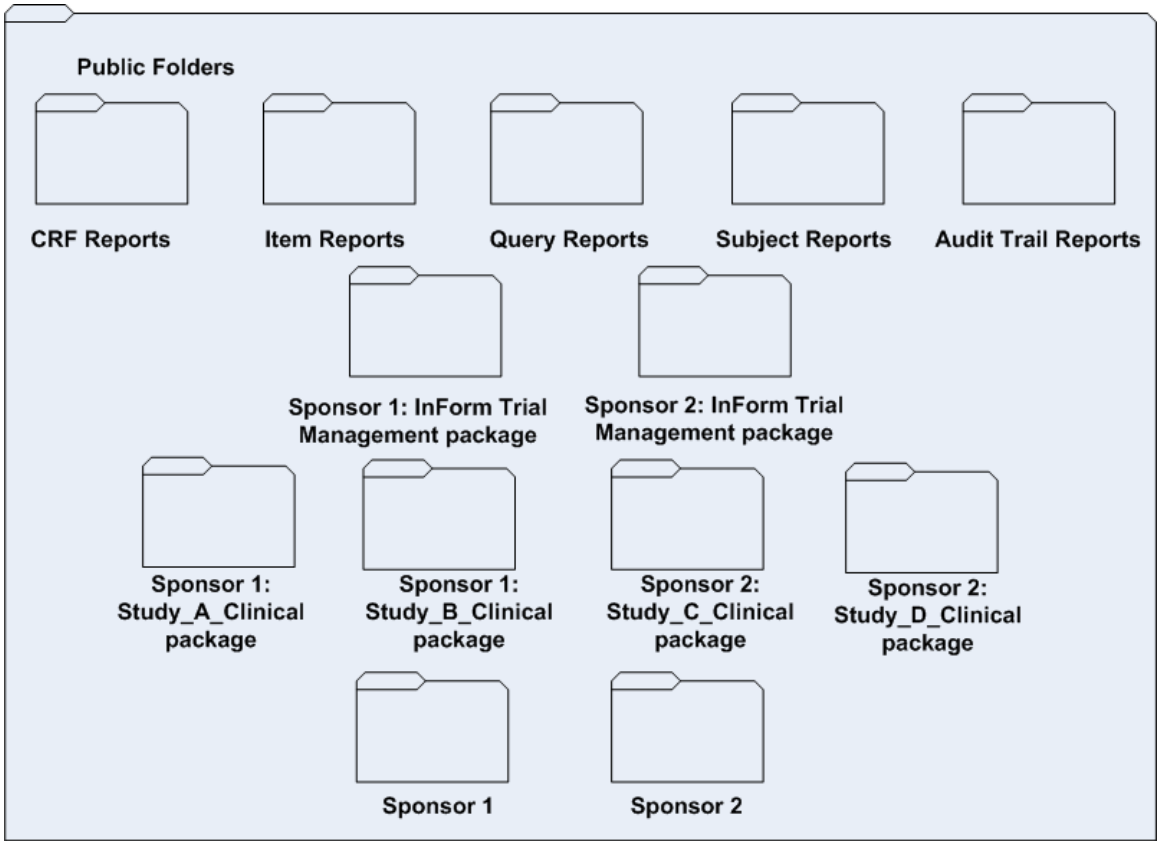
## Folder structure for multiple sponsors, multiple studies

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 the InForm Report Folder Maintenance utility to perform all the configuration steps.

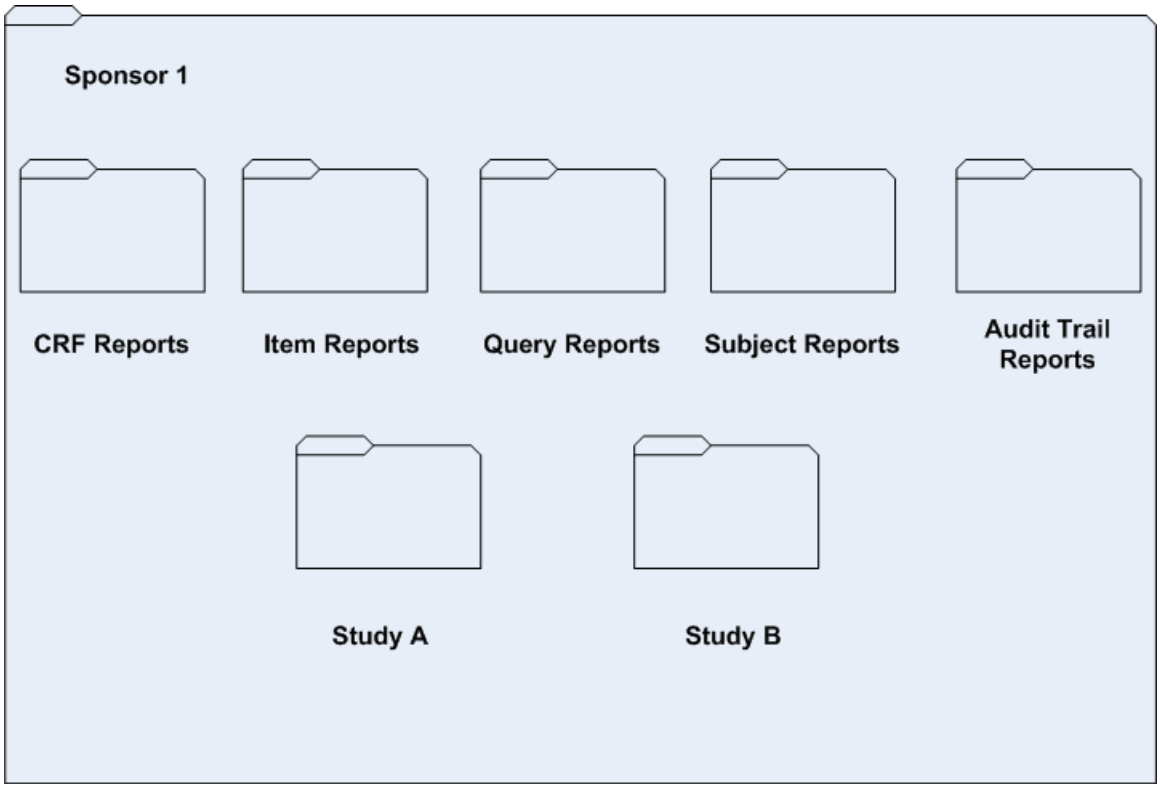
The following illustration shows how the **Public Folders** might look in the Reporting and Analysis portal with folders for two different sponsors, each hosting two different studies.

The following example shows:

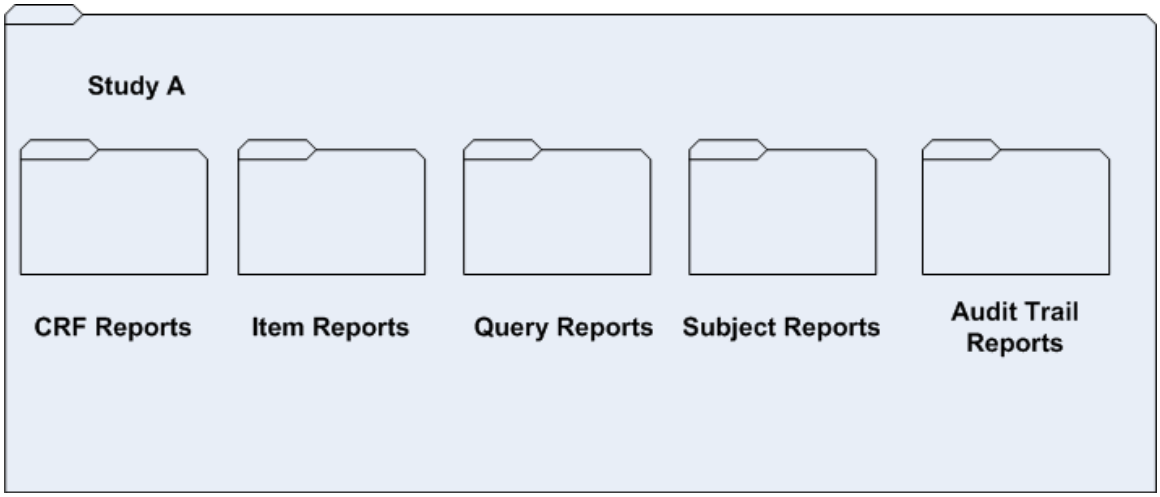
- Four clinical packages: two for Sponsor 1 and two for Sponsor 2.
- Two InForm Trial Management packages: one for Sponsor 1 and one for Sponsor 2.



The contents of each sponsor-specific folder are set up to include the default reporting structure for a single study. The following illustration shows the contents of the Sponsor 1 folder:




The Study_A folder contains all of the default standard report folders.



# Setting up a folder structure for multiple studies or sponsors

## Setting up report packages

All report 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 studies, folders for each of these packages will appear in the **Public Folders** tab in the Reporting and Analysis portal.

### Study-specific clinical package

Each study uses a unique study-specific clinical package. For details on how to publish a study-specific clinical package, see the *Study and Reporting Setup Guide*.

### The InForm Trial Management package

Studies that share a sponsor can share an InForm Trial Management package. This way, any revisions or updates to the package automatically apply to all studies for the sponsor.

You import the InForm Trial Management package by importing the operational model and Reports deployment archive. When you imported this archive as part of your Reporting and Analysis installation, you imported both the standard reports folders and the InForm Trial Management package.

However, to revise packages for individual studies, you must import a study management package for each study.

To import InForm Trial Management packages for different sponsors or studies:

1 Log in to the study as the InForm system administrator.

2 Click **Reports**.

3 Select **Launch**> **Reporting Administration**.

The Administration page appears.

4 Click the **Configuration** tab.

5 Click **Content Administration**.

6 For the **Op package and reports** entry, click **Set properties**.

7 Click the **Import** tab.

8 Select the **InForm Trial Management** checkbox. Make sure that all other options are deselected.

9 Click the pencil icon next to **InForm Trial Management**.

10 Change the name of the InForm Trial Management package to reflect the sponsor or study that will be using the package.

11 Click **OK**.

12 Click **Return**.

The Administration page appears.

13 For the package you created, click the **Run with options - [package name]** icon ( ).

The Run with options - [package name] page appears.

14 In the **Time** section, specify when to run the import.

15 In the **Report Specification upgrade** section, select one of the following:

- Upgrade all report specifications to the latest version.
- Keep the existing report specification versions.

16 Click **Run**.

# Creating new folders for multiple studies or sponsors

## Report package association

Each report is associated with the package (either the InForm Trial Management package or a study-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 study or sponsor who works with the report.

For reports that you created with the InForm Trial Management package:

- **Multiple studies, single sponsor**—You are not required to specify a new package name; the InForm Trial Management package is installed with every study.
- **Multiple studies, 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 study-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 report packages* (on page 386).

The InForm Report Folder Maintenance utility stops when it encounters an 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 study to another, the schema of the clinical portion of the Reporting and Analysis database is unique to each study; therefore, the clinical package that is generated for each study 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? |
|---|---|---|---|
| Study-specific clinical package | Study management data only. | Yes.* | No. |
| Study-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. |
| **Single sponsor, multiple studies** | | | |
| InForm Trial Management package | Study management data only. | No. | No. |
| **Multiple sponsors, multiple studies** | | | |
| InForm Trial Management package | Study 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

1 On the Reporting and Analysis server, double-click the following file:

`\crn\bin\PFMTRSetupUtil.exe`

The InForm Report Folder Maintenance window appears.

2 Enter the following information:

- **User Name** and **Password**—User name and password to log in to the Reporting and Analysis server.
- **LDAP Namespace**—Name of the LDAP namespace that is used for authentication by Sun One Directory Server.
- **Copy from**—Click **Standard report folders** to copy standard report folders, or click **Custom folders** 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 S1 as a prefix for all folders belonging to Study 1, specify S1 in the **Folder prefix** field to instruct the InForm Report Folder Maintenance utility to copy the folders with the S1 prefix.
- **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 Study 2, you might type S2 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 study-specific reports. Since each study clinical package is unique, reports that contain clinical (study-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.

3 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.

**Note:** You can identify the top-level report folder for your company in the **Admin > System Configuration > Reporting user root** field in the user interface. This is the folder you see when you log in to the Reporting and Analysis portal.

APPENDIX A

# Sample data import XML

## In this appendix

# Overview of sample data import XML

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

This sample file below contains the necessary data tags to import screening and enrollment data for subject 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 subject clinical data

The sample file below contains the necessary data tags to import clinical data to subject 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 subject clinical data

The sample file below contains the necessary data tags to edit or clear existing data about a subject.

```
<?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>
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.DISCPULLDOWN.DISCPULLDOWN" VALUE="Not Effective"/>
</PATIENTDATA>

</CLINICALDATA>
```

# Deleting data from an Add Entry itemset

The sample file below contains the necessary data tags to delete an existing Add Entry itemset.

**Note:** You cannot delete or undelete a Repeating Data 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"
SITEMNEMONIC="PF"
FORMSETREFNAME="Visit1"
FORMREFNAME="HH"
SECTIONNAME="PT"
ITEMSETNAME="PT"
DELETEITEMSET="TRUE"
ITEMSETINDEX="4"
REASONOTHER="test reason"
</EDITPATIENTDATA>

</CLINICALDATA>
```

# Undeleting data from an Add Entry itemset

The sample file below contains the necessary data tags to recover a deleted Add Entry itemset.

**Note:** You cannot delete or undelete a Repeating Data 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 subject.

```
<?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 subject records

This example shows the elements in a MedML file that is used to transfer several subjects. Note that each pair of PATIENTSITECHANGE attributes defines current and destination site information for one subject.

```
<?xml version="1.0"?>
<CLINICALDATA>
    <PATIENTSITECHANGE REASON="new subject address">
        <NEWSITE SITEMNEMONIC="MCLEAN"/>
        <CURRENTSITE SITEMNEMONIC="PF" PATIENTNUMBER="1003"/>
    </PATIENTSITECHANGE>
```

<!----For subject 1001 the file specifies a new subject number because a subject already exists at the McLean Hospital site with the same subject number.>

```
    <PATIENTSITECHANGE REASON="new subject address">
        <NEWSITE SITENAME="McLean Hospital" PATIENTNUMBER="1002"/>
        <CURRENTSITE SITEMNEMONIC="PF" PATIENTNUMBER="1001"/>
    </PATIENTSITECHANGE>
```

<!----The DUPLICATEORDER attribute indicates that subject DDD is the second subject with those initials to be screened at the Oracle site.>

```
    <PATIENTSITECHANGE REASON="new subject address">
        <NEWSITE SITEMNEMONIC="MCLEAN"/>
        <CURRENTSITE SITENAME="Phase Forward" PATIENTINITIALS="DDD"
            DUPLICATEORDER="2"/>
    </PATIENTSITECHANGE>
</CLINICALDATA>
```