DOCUMENT NUMBER MTR080368 R3
MITRE TECHNICAL REPORT

MITRE

The Open Checklist Interactive Language (OCIL)

Version 1.1

Maria Casipe, Charles Schmidt
{mcasipe, cmschmidt}@mitre.org

May 2009

Sponsor: NSA I733
Dept. No.: G026
Project No.: 0709N60C-XC

Table of Contents

1 Introduction

The Open Checklist Interactive Language (OCIL) defines a framework for expressing a set of questions to be presented to a user and procedures to interpret responses to these questions for the purpose of developing security checklists. Although its intended domain of use is IT security, its generic nature allows for other applications. For instance, it could be used for authoring research surveys, academic course exams, and instructional walkthroughs.

All organizations must set up their own policies to describe the information to secure, the level of security it needs, and the required state for it to be considered secured. Enforcing and checking compliance with these policies on every system in the organization can be a very daunting task for IT security administrators. For this, they may choose to employ tools designed to automate security compliance checks. Existing automation tools, however, may not be able to completely handle all aspects of these checks. In these situations, IT security administrators face the challenge of manually checking for security policy compliance.

This document describes a standard data model and processing procedure for supporting manual security compliance checks. The data model is designed to encourage portability and reusability of objects. An XML schema of the data model is presented in Section 4.

1.1 Background

There have been a number of initiatives to provide standards for automating security checks. Some of these initiatives include:

- XCCDF (e**X**tensible **C**onfiguration **C**hecklist **D**escription **F**ormat): "*A specification language for writing security checklists, benchmarks, and related kinds of documents."*[1]

- OVAL (**O**pen **V**ulnerability and **A**ssessment **L**anguage): *"OVAL includes a language used to encode system details, and an assortment of content repositories held throughout the community."*[2]

However, none of the existing standards provide language for expressing compliance checks that require manual user interaction. A standard language for expressing manual checks and their results would provide the following benefits:

- Ensure that every manual check is done in the same manner (uniformity) and the required steps are followed, regardless of who performs the check.

- Ensure that manual checks can be tracked for auditing purposes.

[1] Source: http://nvd.nist.gov/xccdf.cfm
[2] Source: http://oval.mitre.org

- Foster development of and interoperability among tools for creating, evaluating, and supporting manual checks.

- Foster collaboration among security-related communities in composing manual checks.

- Enhance manageability of manual checks.

This document describes a data model and processing procedure that provides an extensible, interoperable language for expressing compliance checks that require user feedback.

1.2 Vision for Use

OCIL is designed to enable authors to describe security compliance checks that require user feedback and record their results in an XML document. The end result allows organizations to utilize commercial tools that support manual checks and integrate with existing tools to aid IT security administrators in their tasks. The following scenarios illustrate some of the uses that OCIL will foster.

Scenario 1 –
A security compliance check author writes an IS document that describes a check that requires all labs in a particular building to be secured. This document is made accessible through a web form on the company intranet site. Each security officer accesses this form, which serves as the front-end to an IS interpreter. S/he is asked a series of questions starting with her/his area of responsibility (e.g. what building? what part of the building? etc.). If s/he is responsible for that particular building, then the interpreter asks the next appropriate questions until it produces a result. Otherwise, the evaluation stops. Results are submitted to a repository for further analysis.

Scenario 2 –
A security compliance check author writes an XCCDF document containing checks that needs to be performed for a particular system. However, some of the checks require a complex evaluation that currently cannot be automated. Within the document, s/he includes a reference to an IS document containing a manual check. When an IT security administrator checks for compliance using the XCCDF document, s/he loads it onto an XCCDF interpreter. Upon reaching a manual check, the XCCDF interpreter loads the IS document into the IS interpreter (either within the XCCDF interpreter or an external IS interpreter), which asks the administrator a series of questions. The user's response is collected and interpreted and the result is returned to the XCCDF interpreter which then continues with the remaining checks. The results from both the automated and manual checks are combined into a single report by the XCCDF interpreter.

Scenario 3 –
A company security officer would like to review all compliance checks including those done manually, and produce a report. S/he takes in an IS document that contains the results from the most recent assessment and transforms it into html with an XSL stylesheet.

2 Data model

The data model for OCIL consists of the following high-level object data types:

1. **<ocil>** An OCIL document holds exactly one ocil element. An ocil element is a container for all other elements in the document such as questionnaires, questions, test actions, and results. It also holds metadata describing the document and its creation.

2. **<questionnaire>** A questionnaire represents a series of questions to ask a user that will be evaluated to a single response. Questionnaires are intended to represent a single, discrete check similar to an OVAL definition. Questionnaires can reference one or more test actions or other questionnaires. Questionnaires themselves are referenced through a unique identifier. It also contains descriptive information about the check that is performed by this questionnaire and a property to specify whether it should be treated as a low- or high- priority concept.

3. **<test_action>** A test action defines a test (e.g. question) that needs to be evaluated and the action (i.e. either produce a result or evaluate another test action) to take based on user response. This element is abstract but it has four derived classes which could be used in a document: <boolean_question_test_action>, <choice_question_test_action>, <numeric_question_test_action>, and <string_question_test_action> each expecting a particular data-type for the user's response. The derived types are referenced through a unique identifier.

4. **<question>** A question is an abstract concept with a descriptive text that describes what needs to be answered and a set of instructions on how the user should come to their answer. Question elements are referenced through a unique identifier. There are four derived classes: <boolean_question>, <choice_question>, <numeric_question>, and <string_question> depending on the expected data-type of the user's response.

5. **<results>** The results element holds result information for every questionnaire, test action, and question recorded during a single evaluation of the OCIL document. It also includes metadata about when the evaluation started and ended, descriptive text about the results, and the target system.

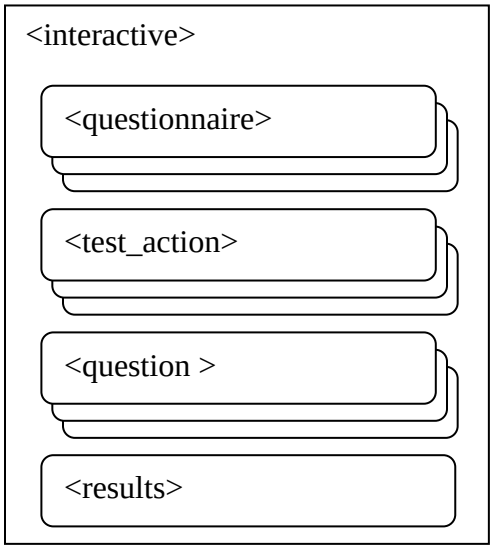

Figure 1. Top-level interactive objects.

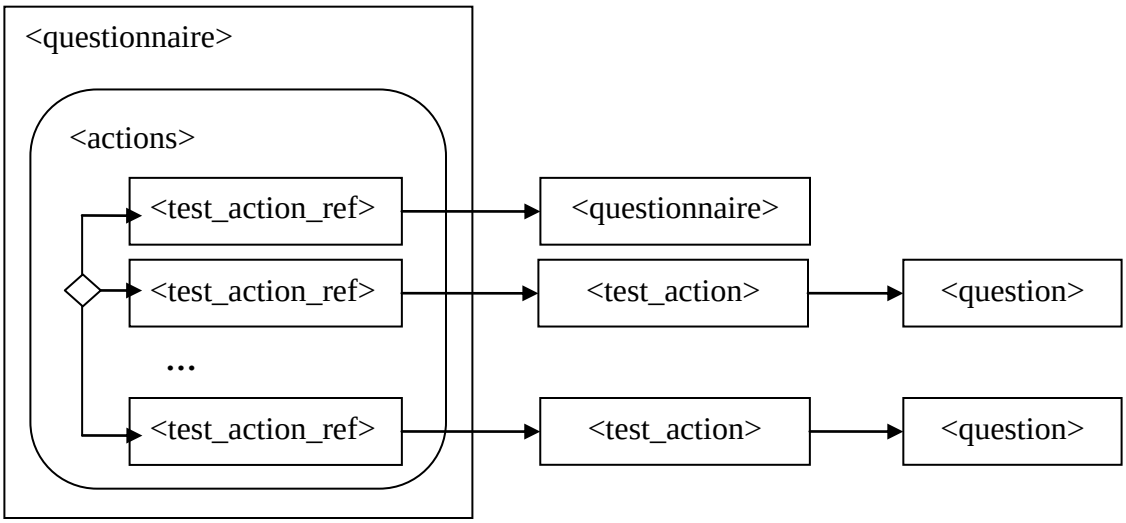

Figure 2. Example of relationships between test actions and questions.

2.1 Element Content Details

The tables below show the properties that make up the main element types of the IS data model.

2.1.1 ocil

Property	Type	Count	Description
scope	FULL or SHORT	0-1	The scope attribute describes how to process evaluation. A value of FULL specifies that evaluation should continue regardless on the status of whether a result can be produced. A value of SHORT indicates

			that evaluation should stop when a result can be computed. (By default: FULL).
generator	GeneratorType	1	A generator describes any information about the creator of the document. Specifically, it includes information about the tool used to generate the document as well as the schema version used and time of creation.
document	DocumentType	0-1	A document includes document-level information that may be presented to a user. Specifically, in includes a title, descriptive information, and notices.
questionnaire	*questionnaire*	1-n	A questionnaire describes a sequence of questions that must be posed to the user in order to determine a state or condition, including processing instructions.
test_action	*test_action*	1-n	A test_action describes a state or condition that must be checked for, including processing instructions.
question	*question*	1-n	A question describes what needs to be posed to a user in order to check for a particular state or condition.
choice_group	*special*	0-n	A choice_group represents a reusable set of choices for a choice_question. Choice_question elements can explicitly specify each of their allowed choices, but this can become tedious if multiple choice questions have overlapping sets of choices. This structure allows a set of choices to be defined once and then reused repeatedly.
results	*results*	0-1	Information about the results on evaluating a questionnaire or test_action.

The ocil element is the root element of an OCIL document. Conceptually, it contains all the questionnaires, test_actions, and questions, and may also contain a results section to store the results of an interaction with the user.

2.1.2 questionnaire

Property	Type	Count	Description
id	questionnaire identifier	1	Unique identifier for a questionnaire element.
priority	PriorityType	0-1	Specifies the priority level of a questionnaire. (By default: LOW).
child_only	boolean	0-1	Specifies whether the questionnaire should be treated as a top-level (=false) or only child-level (=true) questionnaire. (By default: false).
title	TextType	0-1	Descriptive heading or caption that describes the questionnaire.
description	TextType	0-1	Descriptive text that describes the questionnaire in more detail than a title.
reference	ReferenceType	0-n	The reference element contains information about any external references related to this step. Examples could include references to other

			standards such as CVE, CCE, or CPE.
actions	OperationType	1	A set of test_action that must be evaluated for this questionnaire.
notes	string	0-n	Any information related to the questionnaire.

The questionnaire element is the basic unit of an OCIL check. Each questionnaire would correspond to one recommendation or requirement in a security guide. After processing (which includes receiving the necessary responses from a user) a questionnaire will evaluate to one result type (see section 3.1 for a list of result values and their meanings).

The child_only attribute is used to distinguish between a "top-level" and a "low-level" questionnaire. A top-level questionnaire (the default) represents a questionnaire that would be posed to a user directly. Interpreter tools might chose to display a list of all the top-level questionnaires in an OCIL document to a user and allow the user to work through this list. By contrast, a low-level questionnaire is one that was designed to be part of another questionnaire. Specifically, it would be processed and its questions presented to the user as a part of another questionnaire, but would not be displayed independently. An interpreter tool would not wish to display this questionnaire in a list of top-level questionnaires since it represents a sub-section of other questionnaires.

A questionnaire would contain references to one or more test_actions along with an operator used to combine them and an optional negate attribute which could be used to negate the final result. All of this information appears within the questionnaire's <actions> element. Conceptually, this permits an author to pose multiple questions to a user and then aggregate them into a single response. While a single questionnaire may only use a single operator (plus the optional negate) to combine the results of its referenced test_actions, the fact that test_actions can refer to other questionnaires, which can themselves reference multiple test_actions and aggregate them with their own operator allows for arbitrarily complex logical combinations of responses.

2.1.3 test_action

The test_action element is a simple abstract element that serves as the base for a number of child elements. Conceptually, a test_action, as used in an OCIL document to identify a specific question to pose to the user and then handle the user's response. Handling the user response can involve picking a specific result value for this test_action or calling another test_action or questionnaire. (The ResultsChoiceType structure is used to specify this information.) The former case represents a situation where the user's response is conclusive while the latter would be used if the user's response required further follow-up questions in order to reach a final result. The handlers are used to convert a user response (e.g. a number or string) into a result value (e.g. PASS or FAIL). If there is no handler for the given user response, a result value of ERROR is generated.

As noted, the test_action element itself is abstract and would not appear within an OCIL document. It is extended by the question_test_action element, which is also abstract. The question_test_action is then extended by four different child elements, each of which corresponds to a different data-type for the user response. It is these latter elements that would actually be used. The relevant types are described below:

test_action

Property	Type	Count	Description
notes	string	0-n	Used to record information related to the test_action.

question_test_action

Property	Type	Count	Description
id	test_action identifier	1	Unique identifier for a question_test_action.
question_ref	question identifier	1	Identifies a question using its id.
title	TextType	0-1	Descriptive heading or caption that describes the question_test action.
when_unknown	ResultChoiceType	0-1	Processing instruction for when an UNKNOWN value is received.
when_not_tested	ResultChoiceType	0-1	Processing instruction for when a NOT_TESTED value is received.
when_not_applicable	ResultChoiceType	0-1	Processing instruction for when a NOT_APPLICABLE value is received.
when_error	ResultChoiceType	0-1	Processing instruction for when an ERROR value is received.

The question_test_action expands on the test_action element, adding a unique id, a reference to a question, a title, and handler elements for four types of responses to the referenced question.

The four child elements of question_test_action are listed below. In addition to inheriting all the handlers of their parents, they define additional handlers to contain processing instructions for the expected type of user input.

boolean_question_test_action

Property	Type	Count	Description
when_true	ResultChoiceType	1	Processing instruction for when the user responds with TRUE or YES.
when_false	ResultChoiceType	1	Processing instruction for when the user responds with FALSE or NO.

numeric_question_test_action

Property	Type	Count	Description
when_equals	extends ResultChoiceType	0-n	Processing instruction for when the user responds with a specific numeric value. The extension to ResultsChoiceType allows the author to specify this value.
when_range	extends	0-n	Processing instruction for when the user response

	ResultChoiceType		falls within a specified range of values. The extension to ResultsChoiceType allows the author to specify this range.

Note that, despite the fact that either handler may be omitted, at least one handler must appear within the body of a numeric_question_test_action. If multiple handlers could potentially match a particular user response (for example, if there were overlapping ranges) then the first matching handler is used. Since when_equals handlers always come before when_range handlers, this gives when_equals handlers precedence.

string_question_test_action

Property	Type	Count	Description
when_pattern	extends ResultChoiceType	1-n	Processing instruction for when the user responds with a string that matches a specified regular expression. The extension to ResultsChoiceType allows the author to specify this regular expression.

If the user's response could match multiple patterns, the handler with the first matching pattern is used.

choice_question_test_action

Property	Type	Count	Description
when_choice	extends ResultChoiceType	1-n	Processing instruction for when the user responds with one of a specified list of choices. The extension to ResultsChoiceType allows the author to specify multiple choices.

Choice questions present the user with a list of possible responses and the user selects one of them. The author should create handlers to cover all of the possible choices a user might select.

2.1.4 question

Property	Type	Count	Description
id	question identifier	0-1	Unique identifier for a question.
question_text	string	1-n	Descriptive text to be posed to the user as a question.
instructions	*special*	0-1	A sequence of steps intended to guide the user in determining an answer to a question.
notes	string	0-n	Any information related to the question.

A question element represents a question to pose to the user and an optional set of instructions for the user to follow in order to arrive at their answer. The question element itself is abstract and does not appear in OCIL documents. Instead, the four child elements, one for each data type of user response are used.

The child elements all inherit the constructs of their parent element. In addition, all four may contain an optional attribute to designate a default value. Two child element types contain

additional structure beyond this default value: The boolean_question element also contains a mode attribute to indicate to an interpreter whether the user should pick between TRUE/FALSE or YES/NO. The choice_question element contains a list of child elements, which specify one possible choice for the user's response, as well as references to child_group elements, which describe a list of choices. Choice elements and choice_group references may be interleaved and are presented to the user in the order in which they appear.

2.1.5 results

Property	Type	Count	Description
start_time	dateTime	0-1	Specifies when the evaluation of the OCIL document started.
end_time	dateTime	0-1	Specifies when the evaluation of the OCIL document completed.
title	TextType	0-1	A descriptive heading or caption that describes the result set.
target	string	0-n	Host name of the system(s) being checked.
target_address	string	0-n	Address(es) of the system(s) being checked.
questionnaire_result	*special*	0-n	Describes the result of evaluating a questionnaire.
question_result	*special*	0-n	Describes the result of evaluating a question.
test_action_result	*special*	0-n	Describes the result of evaluating a test_action.

The results element is used to store the results from processing the document. These results include not only the evaluated results of each questionnaire, but a breakdown of the results for each test_action and the user's response to each question.

3 Requirements

This section describes the processing requirements that an interpreter must follow in order to correctly process an OCIL file.

3.1 Types of Results

The result value of a questionnaire or a test action can be any one of the following:

1. **PASS**. The state or condition being tested is achieved or satisfied.
2. **FAIL**. The state or condition being tested is not achieved or satisfied.
3. **UNKNOWN**. The sate or condition being tested could not be determined.
4. **ERROR**. The user answered with an unacceptable or unhandled value; OR the interpreter encountered an unhandled situation or system error.

5. **NOT_APPLICABLE**. The questionnaire or test action does not apply to the goal as determined by user response(s).

6. **NOT_TESTED**. The questionnaire or test action has not been inspected by the user for the following reasons: (a) the user marked a question referenced by a test action as not tested, or (b) the questionnaire or question referenced by a test action has not been presented to the user yet.

3.2 Scope of Evaluation

Evaluation can be controlled through the @scope attribute of the <ocil> element. It can either have a value of SHORT or FULL. By default, it is set to FULL.

Setting the value of the @scope attribute to SHORT forces the evaluation of any questionnaire to stop when sufficient information has been gathered to produce a final result. For example, suppose there are 10 test actions in a questionnaire and based on the user's response on the 5th test action, the result of the questionnaire can be computed. At this point, the evaluation will stop if @scope is set to SHORT.

Setting the value of the @scope attribute to FULL forces questionnaires to evaluate all test actions even if the final result will be unaffected by the subsequent questions. In the example above, if the @scope was set to FULL, evaluation would finish when the 10th question has been answered even though the final result was known after the evaluation of the 5th test action.

3.3 Evaluating Test Actions

The IS defines a structure called test action. A test action defines what needs to be tested (e.g. a question), and what action to take based on user response. An action can either be an event that triggers the next test action to be evaluated or it can simply produce a result. If the action is to produce a result, then the result is propagated up to its calling test action.

A questionnaire may contain multiple references to other test actions. To evaluate a questionnaire, each referenced test action must be evaluated, modified by the @scope attribute in the <ocil> element as described in section 2.1. The results of the referenced test actions are combined to produce the final result of the questionnaire. The following steps describe how the results are combined (in order):

1. The value of the @operation attribute in the <actions> element is applied. This attribute can either have an AND or OR value. By default, its value is set to AND. The truth table below (see Table 1) defines how to combine results.

2. The value of the @negate attribute in the <actions> element is applied. This attribute can either have a true or false value. By default, its value is set to false. When set to true, the result returned by the questionnaire is changed in the following way: FAIL becomes PASS, PASS becomes FAIL, and all other results are unchanged.

Table 1. Truth Table for Combining Test Action Individual Results

Operator	Number of Individual Results						Final Result
	PASS	FAIL	ERROR	UNKNOWN	NOT TESTED	NOT APPLICABLE	
AND	1+	0	0	0	0	0+	PASS
	0+	1+	0+	0+	0+	0+	FAIL
	0+	0	1+	0+	0+	0+	ERROR
	0+	0	0	1+	0+	0+	UNKNOWN
	0+	0	0	0	1+	0+	NOT TESTED
	0	0	0	0	0	1+	NOT APPLICABLE
	0	0	0	0	0	0	NOT TESTED
OR	1+	0+	0+	0+	0+	0+	PASS
	0	1+	0	0	0	0+	FAIL
	0	0+	1+	0+	0+	0+	ERROR
	0	0+	0	1+	0+	0+	UNKNOWN
	0	0+	0	0	1+	0+	NOT TESTED
	0	0	0	0	0	1+	NOT APPLICABLE
	0	0	0	0	0	0	NOT TESTED

3.4 Test Actions and Error Processing

As mentioned in Section 2.2, a test action defines what needs to be tested (e.g. a question), and what action to take based on user response. Every test action that references a question may contain any or all of the following elements: <when_error>, <when_unknown>, <when_not_tested>, and <when_not_applicable>. The names suggest the condition when the action defined within the element is to be applied:

- <when_error> defines the action to take when an error occurs.
- <when_unknown> defines the action to take when the user marks a question as having an unknown answer.
- <when_not_tested> defines the action to take when the user decides not to evaluate a test.
- <when_not_applicable> defines the action to take when the user marks a question as not applicable.

The above elements are optional, i.e., some or all of them may not exist for a particular test action.

The IS contains structures for different question types depending on the expected data-type of the user's response. For each type of question, there is an associated variant of the test action element that contains additional handlers to process the anticipated user response. For instance,

for a <boolean_test_action>, a <when_true> and <when_false> are added to define what actions to take when a test evaluates to a true (or yes) or false (or no), respectively.

When a test evaluates to a value with no defined action, the result is an ERROR. For example, if a user marks a test as NOT_APPLICABLE, but there is no <when_not_applicable> handler, the <test_action> will evaluate to ERROR. Likewise, if a user provides a normal answer (boolean, number, string, etc.), but there is no handler for that answer this will also evaluate to ERROR. For example, if the user returns 9 but there is no <when_equals> or <when_range> handlers that match a value of 9 then an ERROR is returned. This means that, apart from an ERROR result, the only time any other result values are returned by a test action would be if the handler explicitly provided a return value.

Consider the following example:
(Source: ISO IEC 27002 2005 Information Security Audit Tool)

```
8.1 Question 1.  Have you reduced the risk of theft, fraud, or misuse of
facilities by making sure that all prospective employees understand their
responsibilities before you hire them?  YES  NO N/A
```

The above question requires an answer of YES, NO, or N/A (Not Applicable). It is best modeled with a <boolean_question> element. For instance,

```
<boolean_question id="inter:mitre.org:question:1" model="MODEL_YES_NO">
   <question_text>
      Have you reduced the risk of theft, fraud, or misuse of facilities by
      making sure that all prospective employees understand their
      responsibilities before you hire them?
   </question_text>
</boolean_question>
```

To describe what happens when a user responds to this type of question, a <boolean_question_test_action> can be defined in the following manner:

```
<boolean_question_test_action id="inter:mitre.org:testaction:1"
   question_ref="inter:mitre.org:question:1">
   <when_true>
      <result>PASS</result>
   </when_true>
   <when_false>
      <test_action_ref>inter:mitre.org:testaction:2</test_action_ref>
   </when_false>
   <when_not_applicable>
      <test_action_ref>inter:mitre.org:testaction:3</test_action_ref>
   </when_not_applicable>
</boolean_question_test_action>
```

An interpreter would present the referenced question to the user. Based on the user's response to the question, the test action would perform different actions. Specifically, if the user responds:

YES	The <when_true> handler is invoked, which sets the result of this test action to PASS.
NO	The <when_false> handler is invoked. This causes the interpreter to evaluate the test action with id *inter:mitre.org:testaction:2*. Whatever that test action evaluates to becomes the result of this test action.

N/A	The <when_not_applicable> handler is invoked. This causes the interpreter to evaluate the test action with id *inter:mitre.org:testaction:3* Whatever that test action evaluates to becomes the result of this test action.
Anything else	There is no handler for any other responses so other responses set the result of this test action to ERROR

Note that for simplicity purposes, 'yes' and 'true' responses are mapped to a <when_true> element. Similarly, 'no' and 'false' responses are mapped to a <when_false> element.

A <numeric_test_action> is a test action that references a <numeric_question>. It contains a set of <when_equals> and/or <when_range> element. A <when_equals> element defines what action to take when a particular value matches the response of the user. Similarly, a <when_range> element defines what action to take when the user's response is within a specified range of values, for example [29,100], [101,132), or (131, 249]. If the value given by the user matches multiple conditions, then the first matching handler is applied. Since <when_equals> handlers always come before <when_range> preference is given to exact matches.

A <string_test_action> is a test action that references a <string_question>. It contains a set of <when_pattern> element. A <when_pattern> element defines what action to taken when the user's response matches a particular regular expression. Similar to <numeric_test_action>, if the value given by the user matches multiple conditions, then the first <when_pattern> matched is applied.

A <choice_test_action> is a test action that references a <choice_question>. It contains a set of <when_choice> element. A <when_choice> element defines what action to take when the user's responds matches one of a list of choices. The schema prevents multiple <when_choice> elements from containing references to the same choice within a single <choice_test_action>.

4 Example

```
<?xml version="1.0" encoding="UTF-8"?>
<ocil xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.mitre.org/ocil/1.1 ocil.xsd"
    xmlns="http://www.mitre.org/ocil/1.1">

    <!--Generator -->
    <generator>
        <schema_version>1.1</schema_version>
        <timestamp>2009-5-20T00:00:00</timestamp>
    </generator>

    <!--Questionnaires -->
    <questionnaire id="ocil:mitre.org:questionnaire:1" priority="LOW">
        <title>Physical security Requirements</title>
        <description> Inadequate physical protection can undermine all other security precautions
            utilized to protect the system. This can jeopardize the confidentiality,
            availability, and integrity of the system. Physical security of the AIS is the first
            line protection of any system. Note: Critical servers should be located in rooms, or
            locked cabinets, that are accessible only to authorized systems personnel. User
            workstations containing sensitive data should be in access controlled areas.
        </description>
        <actions priority="LOW">
            <test_action_ref priority="LOW">ocil:mitre.org:testaction:11</test_action_ref>
        </actions>
    </questionnaire>
```

```
<questionnaire id="ocil:mitre.org:questionnaire:2" priority="LOW">
    <title>Users with Administrative Privileges</title>
    <description> Using a privileged account to perform routine functions makes the computer
        vulnerable to attack by any virus or Trojan Horse inadvertently introduced during a
        session that has been granted full privileges. The rule of least privilege should
        always be enforced. </description>
    <actions priority="LOW" operation="OR">
        <test_action_ref priority="LOW">ocil:mitre.org:testaction:21</test_action_ref>
        <test_action_ref priority="LOW">ocil:mitre.org:testaction:22</test_action_ref>
    </actions>
</questionnaire>

<questionnaire id="ocil:mitre.org:questionnaire:3" priority="LOW">
    <title>Backup Administrator Account</title>
    <description> Backup Operators are able to read and write to any file in the system,
        regardless of the rights assigned to it. Backup and restore rights permit users to
        circumvent the file access restrictions present on NTFS disk drives for the purpose
        of backup and restore. Members of the Backup Operators group should have special
        logon accounts for performing their backup duties. </description>
    <actions priority="LOW" operation="AND">
        <test_action_ref priority="LOW">ocil:mitre.org:testaction:31</test_action_ref>
        <test_action_ref priority="LOW">ocil:mitre.org:testaction:32</test_action_ref>
    </actions>
</questionnaire>

<!--Test Actions -->
<boolean_question_test_action id="ocil:mitre.org:testaction:11"
    question_ref="ocil:mitre.org:question:11">
    <when_true>
        <result>PASS</result>
    </when_true>
    <when_false>
        <result>FAIL</result>
    </when_false>
</boolean_question_test_action>
<choice_question_test_action id="ocil:mitre.org:testaction:21"
    question_ref="ocil:mitre.org:question:21">
    <when_choice>
        <result>PASS</result>
        <choice_ref>ocil:mitre.org:choice:211</choice_ref>
    </when_choice>
    <when_choice>
        <result>FAIL</result>
        <choice_ref>ocil:mitre.org:choice:212</choice_ref>
    </when_choice>
</choice_question_test_action>

<choice_question_test_action id="ocil:mitre.org:testaction:22"
    question_ref="ocil:mitre.org:question:22">
    <when_choice>
        <result>FAIL</result>
        <choice_ref>ocil:mitre.org:choice:221</choice_ref>
    </when_choice>
    <when_choice>
        <result>PASS</result>
        <choice_ref>ocil:mitre.org:choice:222</choice_ref>
    </when_choice>
</choice_question_test_action>

<numeric_question_test_action question_ref="ocil:mitre.org:question:31"
    id="ocil:mitre.org:testaction:31">
    <when_range>
        <result>FAIL</result>
        <range>
            <min>0</min>
            <max>10</max>
        </range>
    </when_range>
    <when_range>
        <result>PASS</result>
        <range>
```

```
                <min>11</min>
                <max>100</max>
            </range>
        </when_range>
    </numeric_question_test_action>

    <string_question_test_action question_ref="ocil:mitre.org:question:32"
        id="ocil:mitre.org:testaction:32">
        <when_pattern>
            <result>PASS</result>
            <pattern>secured</pattern>
        </when_pattern>
        <when_pattern>
            <result>FAIL</result>
            <pattern>*</pattern>
        </when_pattern>
    </string_question_test_action>

    <!--Questions -->
    <boolean_question id="ocil:mitre.org:question:11" model="MODEL_YES_NO">
        <question_text>Has equipment been relocated to a controlled access area?</question_text>
    </boolean_question>

    <choice_question default_answer_ref="ocil:mitre.org:choice:211"
         id="ocil:mitre.org:question:21">
        <question_text>Which one of the following is true?</question_text>
        <choice id="ocil:mitre.org:choice:211"> All administrators have a separate account for
            normal user tasks. </choice>
        <choice id="ocil:mitre.org:choice:212"> Not all administrators have separate account for
            normal user tasks. </choice>
    </choice_question>
    <choice_question default_answer_ref="ocil:mitre.org:choice:221"
         id="ocil:mitre.org:question:22">
        <question_text>Which one are you using for system administration?</question_text>
        <choice id="ocil:mitre.org:choice:221"> I'm using my normal user account. </choice>
        <choice id="ocil:mitre.org:choice:222"> I'm using the built-in system administrator
            account.
        </choice>
    </choice_question>

    <numeric_question id="ocil:mitre.org:question:31">
        <question_text> How many user ids do each Backup Operator have for performing backup
            duties?
        </question_text>
    </numeric_question>

    <string_question id="ocil:mitre.org:question:32">
        <question_text> IAO stored details about the backup administrator account in a _______
            location. </question_text>
    </string_question>
</ocil>
```

5 Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
    DOCUMENT:     ocil.xsd
    CREATED ON:   7 February 2005 (Interactive Schema)
    LAST UPDATED ON: May 20, 2009 (OCIL)
    AUTHORS:      David Waltermire, Jon Baker, Maria Casipe, Charles Schmidt
    VERSION:      1.1

    DESCRIPTION:  XML Schema for defining interactive questions to be used as external checks
-->
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:inter="http://www.mitre.org/ocil/1.1"
   xmlns:xml="http://www.w3.org/XML/1998/namespace"
   targetNamespace="http://www.mitre.org/ocil/1.1"
   elementFormDefault="qualified" attributeFormDefault="unqualified">
```

```
<xsd:annotation>
   <xsd:documentation>VERSION 1.1</xsd:documentation>
   <xsd:documentation> The Open Checklist Interactive Language (OCIL) is a language to express
      a set of questions to be presented to a user and procedures to interpret responses to
      these questions for the purpose of developing security checklists. Although its intended
      domain of use is IT security, its generic nature allows for other applications. For
      instance, it could be used for authoring research surveys, academic course exams, and
      instructional walkthroughs. </xsd:documentation>
   <xsd:documentation> This document was originally developed by David Waltermire (The Center
      for Internet Security) and has been revised by The MITRE Corp with input from the
      security benchmark community. It is intended for developers and assumes familiarity with
      XML. </xsd:documentation>
</xsd:annotation>

<xsd:import namespace="http://www.w3.org/XML/1998/namespace"
   schemaLocation="http://www.w3.org/2001/03/xml.xsd">
   <xsd:annotation>
      <xsd:documentation> This namespace is required for xml:lang support.
      </xsd:documentation>
   </xsd:annotation>
</xsd:import>

<!-- ************************************************************************** -->
<!-- *   ocil (Root) Element                                                  * -->
<!-- ************************************************************************** -->
<xsd:element name="ocil">
   <xsd:annotation>
      <xsd:documentation> The ocil element is the root XML element of an OCIL document. It
         contains information about one or more questionnaires. It may also contain results
         elements to store prior responses. </xsd:documentation>
   </xsd:annotation>
   <xsd:complexType>
      <xsd:sequence>
         <xsd:element name="generator" type="inter:GeneratorType" minOccurs="1" maxOccurs="1">
            <xsd:annotation>
               <xsd:documentation>The generator element contains information related to the
                  generation of the file. Specifically, a generator contains information
                  about the application used to create the file, when it was created, and
                  the schema to use to validate it. </xsd:documentation>
            </xsd:annotation>
         </xsd:element>
         <xsd:element name="document" type="inter:DocumentType" minOccurs="0" maxOccurs="1">
            <xsd:annotation>
               <xsd:documentation> This element contains document-level information,
                  including title, descriptions, and notices. </xsd:documentation>
            </xsd:annotation>
         </xsd:element>
         <xsd:element ref="inter:questionnaire" minOccurs="1" maxOccurs="unbounded">
            <xsd:annotation>
               <xsd:documentation>A questionnaire contains a set of questions that
                  determines compliance with a check. Each questionnaire returns a value
                  based on the responses to the various questions that it references. Each
                  questionnaire should represent a single compliance check, such as might
                  be referenced by an XCCDF Rule.</xsd:documentation>
            </xsd:annotation>
         </xsd:element>
         <xsd:element ref="inter:test_action" minOccurs="1" maxOccurs="unbounded">
            <xsd:annotation>
               <xsd:documentation>The test_action element contains information about what
                  action to take based on the answer to a referenced question element
                  within a questionnaire. It can be a compound_test_action,
                  boolean_question_test_action, choice_question_test_action,
                  numeric_question_test_action, or string_question_test_action.
               </xsd:documentation>
            </xsd:annotation>
         </xsd:element>
         <xsd:element ref="inter:question" minOccurs="1" maxOccurs="unbounded">
            <xsd:annotation>
               <xsd:documentation>The question element contains information for a single
                  question to be answered. Based on the data type of acceptable answers to
                  the question, it can be a boolean_question, choice_question,
```

```
                  numeric_question, or string_question. </xsd:documentation>
            </xsd:annotation>
         </xsd:element>
         <xsd:element ref="inter:choice_group" minOccurs="0" maxOccurs="unbounded">
            <xsd:annotation>
               <xsd:documentation> Holds choice groups which represent possible sets of
                  choices for choice_questions. Choice_groups may be reused across
                  multiple choice_questions. </xsd:documentation>
            </xsd:annotation>
         </xsd:element>
         <xsd:element ref="inter:results" minOccurs="0" maxOccurs="1">
            <xsd:annotation>
               <xsd:documentation>The results element contains the results of an evaluation
                  of the OCIL file. This includes records of all questionnaire results,
                  question results, and test_action results.</xsd:documentation>
            </xsd:annotation>
         </xsd:element>
      </xsd:sequence>
      <xsd:attribute name="scope" use="optional" default="FULL" type="inter:ScopeType">
         <xsd:annotation>
            <xsd:documentation>This specifies the test_actions referenced by the
               questionnaires in this document should be evaluated. If a value of FULL is
               provided, then all test_actions in a questionnaire will be evaluated
               regardless of whether the result can be known ahead of time. If a value of
               SHORT is provided a questionnaire will stop evaluating test_actions as soon
               as the final result has been determined. (For example, if one test_action
               evaluates to FAIL and the test_actions of a questionnaire are being ANDed
               together, the result is immediately known to be FAIL regardless of the
               results of other test_actions. If the scope is FULL, even though it is known
               that the questionnaire will return FAIL, the other test_actions will still
               be evaluated. If the scope is SHORT, no further test_actions would be
               evaluated since the final result had been determined.)</xsd:documentation>
         </xsd:annotation>
      </xsd:attribute>
   </xsd:complexType>

   <!-- ************************************************************************** -->
   <!-- *  Questionnaire  references                                             * -->
   <!-- ************************************************************************** -->
   <xsd:key name="questionnaireIdKey">
      <xsd:selector xpath="inter:questionnaire"/>
      <xsd:field xpath="@id"/>
   </xsd:key>
   <xsd:keyref refer="inter:questionnaireIdKey" name="questionnaireKeyRef">
      <xsd:selector xpath=".//inter:questionnaire_result"/>
      <xsd:field xpath="@questionnaire_ref"/>
   </xsd:keyref>

   <!-- ************************************************************************** -->
   <!-- *  Test Action  references                                               * -->
   <!-- ************************************************************************** -->
   <xsd:key name="testActionIdKey">
      <xsd:annotation>
         <xsd:documentation>test_actions include question_test_action and
            compound_test_action (Questionnaires are CompoundTestAction)</xsd:documentation>
      </xsd:annotation>
      <xsd:selector

   xpath="inter:boolean_question_test_action|inter:choice_question_test_action|inter:numeric_ques
tion_test_action|inter:string_question_test_action|inter:questionnaire"/>
      <xsd:field xpath="@id"/>
   </xsd:key>
   <xsd:keyref refer="inter:testActionIdKey" name="testActionKeyRef">
      <xsd:selector xpath=".//inter:test_action_ref"/>
      <xsd:field xpath="."/>
   </xsd:keyref>
   <xsd:keyref refer="inter:testActionIdKey" name="testActionResultKeyRef">
      <xsd:selector xpath=".//inter:test_action_result"/>
      <xsd:field xpath="@test_action_ref"/>
   </xsd:keyref>
```

```
   <!-- ************************************************************************** -->
   <!-- *  Question references                                                   * -->
   <!-- ************************************************************************** -->
   <xsd:key name="booleanQuestionIdKey">
      <xsd:selector xpath="inter:boolean_question"/>
      <xsd:field xpath="@id"/>
   </xsd:key>
   <xsd:keyref refer="inter:booleanQuestionIdKey" name="booleanQuestionTestActionKeyRef">
      <xsd:selector xpath="inter:boolean_question_test_action|inter:boolean_question_result"/>
      <xsd:field xpath="@question_ref"/>
   </xsd:keyref>
   <xsd:key name="choiceQuestionIdKey">
      <xsd:selector xpath="inter:choice_question"/>
      <xsd:field xpath="@id"/>
   </xsd:key>
   <xsd:keyref refer="inter:choiceQuestionIdKey" name="choiceQuestionTestActionKeyRef">
      <xsd:selector xpath="inter:choice_question_test_action|inter:choice_question_result"/>
      <xsd:field xpath="@question_ref"/>
   </xsd:keyref>
   <xsd:key name="numericQuestionIdKey">
      <xsd:selector xpath="inter:numeric_question"/>
      <xsd:field xpath="@id"/>
   </xsd:key>
   <xsd:keyref refer="inter:numericQuestionIdKey" name="numericQuestionTestActionKeyRef">
      <xsd:selector xpath="inter:numeric_question_test_action|inter:numeric_question_result"/>
      <xsd:field xpath="@question_ref"/>
   </xsd:keyref>
   <xsd:key name="stringQuestionIdKey">
      <xsd:selector xpath="inter:string_question"/>
      <xsd:field xpath="@id"/>
   </xsd:key>
   <xsd:keyref refer="inter:stringQuestionIdKey" name="stringQuestionTestActionKeyRef">
      <xsd:selector xpath="inter:string_question_test_action|inter:string_question_result"/>
      <xsd:field xpath="@question_ref"/>
   </xsd:keyref>

   <!-- ************************************************************************** -->
   <!-- *  Choice and choice_group reference keys                                * -->
   <!-- ************************************************************************** -->
   <xsd:key name="choiceIdKey">
      <xsd:selector xpath=".//inter:choice"/>
      <xsd:field xpath="@id"/>
   </xsd:key>
   <xsd:keyref refer="inter:choiceIdKey" name="choiceIdKeyRef">
      <xsd:selector xpath=".//inter:when_choice/inter:choice_ref"/>
      <xsd:field xpath="."/>
   </xsd:keyref>
   <xsd:keyref refer="inter:choiceIdKey" name="defaultAnswerKeyRef">
      <xsd:selector xpath=".//inter:choice_question"/>
      <xsd:field xpath="@default_answer_ref"/>
   </xsd:keyref>
   <xsd:key name="choiceGroupIdKey">
      <xsd:selector xpath=".//inter:choice_group"/>
      <xsd:field xpath="@id"/>
   </xsd:key>
   <xsd:keyref refer="inter:choiceGroupIdKey" name="choiceGroupIdKeyRef">
      <xsd:selector xpath=".//inter:choice_question/inter:choice_group_ref"/>
      <xsd:field xpath="."/>
   </xsd:keyref>
</xsd:element>


<!-- ************************************************************************** -->
<!-- *   Questionnaire Element                                                * -->
<!-- ************************************************************************** -->
<xsd:element name="questionnaire">
   <xsd:annotation>
      <xsd:documentation>A questionnaire represents specific question or set of questions that
         evaluate to a single result. A questionnaire may contain multiple test_actions.
         test_actions may be nested and aggregated thru some acceptable operation to produce
         the result of a check. </xsd:documentation>
```

```
   </xsd:annotation>
   <xsd:complexType>
      <xsd:complexContent>
         <xsd:extension base="inter:CompoundTestActionType">
            <xsd:attribute name="id" type="inter:QuestionnaireIDPattern" use="required">
               <xsd:annotation>
                  <xsd:documentation> Each questionnaire is required to have a unique
                     identifier that conforms to the definition of NCName in the
                     Recommendation "Namespaces in XML 1.0", i.e., all XML 1.0 names that
                     does not contain colons. </xsd:documentation>
               </xsd:annotation>
            </xsd:attribute>
            <xsd:attribute name="priority" type="inter:PriorityType" use="optional"
               default="LOW">
               <xsd:annotation>
                  <xsd:documentation> Priority is an optional attribute that can either be
                     HIGH, MEDIUM or LOW. It specifies the importance of a paritcular
                     test action reference. </xsd:documentation>
               </xsd:annotation>
            </xsd:attribute>
            <xsd:attribute name="child_only" type="xsd:boolean" use="optional"
               default="false">
               <xsd:annotation>
                  <xsd:documentation>This attribute specifies whether or not this
                     questionnaire should only appear as a child of another
                     questionnaire. All questionnaires must be defined within the body of
                     the ocil element and, by default, interpreters might simply grab all
                     questionnaires and present them to a user. However, questionnaires
                     can reference other questionnaires through a test_action_ref. If an
                     author references a questionnaire in this way they may not wish that
                     the questionnaire be presented to a user except as a child of
                     another questionnaire. By setting the child_only attribute to true,
                     the author is indicating that the given questionnaire should not be
                     a "top-level" questionnaire but should instead only be presented as
                     the child of another questionnaire.</xsd:documentation>
               </xsd:annotation>
            </xsd:attribute>
         </xsd:extension>
      </xsd:complexContent>
   </xsd:complexType>
</xsd:element>

<!-- ************************************************************************** -->
<!-- *   Question Elements                                                    * -->
<!-- ************************************************************************** -->
<xsd:element name="question" type="inter:QuestionType" abstract="true">
   <xsd:annotation>
      <xsd:documentation> A question elements contains information one question that needs to
         be answered by a user. It can be a boolean_question, choice_question,
         numeric_question, or string_question depending on the set of acceptable answers.
      </xsd:documentation>
   </xsd:annotation>
</xsd:element>
<xsd:element name="boolean_question" substitutionGroup="inter:question">
   <xsd:annotation>
      <xsd:documentation> A boolean_question is a type of question with valid responses of
         either TRUE, FALSE or YES, NO. </xsd:documentation>
   </xsd:annotation>
   <xsd:complexType>
      <xsd:complexContent>
         <xsd:extension base="inter:QuestionType">
            <xsd:attribute name="default_answer" type="xsd:boolean" use="optional">
               <xsd:annotation>
                  <xsd:documentation> The default_answer attribute specifies the default
                     value of the boolean_question. Its value may be set to true or
                     false.</xsd:documentation>
               </xsd:annotation>
            </xsd:attribute>
            <xsd:attribute name="model" default="MODEL_YES_NO" use="optional"
               type="inter:BooleanQuestionModelType">
               <xsd:annotation>
```

```
                  <xsd:documentation> The model attribute specifies whether to the user
                     should respond with True, False or YES, NO. If the value of this
                     attribute is not set, then it defaults to MODEL_YES_NO (i.e.
                     response can either be YES or NO). </xsd:documentation>
               </xsd:annotation>
            </xsd:attribute>
         </xsd:extension>
      </xsd:complexContent>
   </xsd:complexType>
</xsd:element>
<xsd:element name="choice_question" substitutionGroup="inter:question">
   <xsd:annotation>
      <xsd:documentation> A choice_question is a type of question element with one or more
         acceptable answers specified by the author. The user will select one of these
         specified answers as their response. Acceptable answers are specified either
         explicitly using the choice element or implicitly using the choice_group_ref element
         to reference a choice_group element. Choices are presented in the order in which
         they are provided. All the choices in a choice_group are inserted in the order in
         which they appear within the choice_group. </xsd:documentation>
   </xsd:annotation>
   <xsd:complexType>
      <xsd:complexContent>
         <xsd:extension base="inter:QuestionType">
            <xsd:choice minOccurs="1" maxOccurs="unbounded">
               <xsd:element ref="inter:choice">
                  <xsd:annotation>
                     <xsd:documentation> Holds the information associated with one of the
                        possible responses to this choice_question. </xsd:documentation>
                  </xsd:annotation>
               </xsd:element>
               <xsd:element name="choice_group_ref" type="inter:ChoiceGroupIDPattern">
                  <xsd:annotation>
                     <xsd:documentation> Holds a reference to a choice_group. The
                        questions described in this choice group are used as possible
                        responses for this choice_question. </xsd:documentation>
                  </xsd:annotation>
               </xsd:element>
            </xsd:choice>
            <xsd:attribute name="default_answer_ref" type="inter:ChoiceIDPattern"
               use="optional">
               <xsd:annotation>
                  <xsd:documentation> The default_answer_ref specifies the choice id of
                     the default answer to the question. </xsd:documentation>
               </xsd:annotation>
            </xsd:attribute>
         </xsd:extension>
      </xsd:complexContent>
   </xsd:complexType>
   <xsd:unique name="choiceGroupIdUniqueInQuestion">
      <xsd:selector xpath="inter:choice_group_ref"/>
      <xsd:field xpath="."/>
   </xsd:unique>
</xsd:element>
<xsd:element name="numeric_question" substitutionGroup="inter:question">
   <xsd:annotation>
      <xsd:documentation> A numeric_question is a type of question_element that requires a
         numeric answer. Acceptable values may be positive or negative and may include
         decimals.</xsd:documentation>
   </xsd:annotation>
   <xsd:complexType>
      <xsd:complexContent>
         <xsd:extension base="inter:QuestionType">
            <xsd:attribute name="default_answer" type="xsd:decimal" use="optional">
               <xsd:annotation>
                  <xsd:documentation> An optional default value may be specified as the
                     answer. </xsd:documentation>
               </xsd:annotation>
            </xsd:attribute>
         </xsd:extension>
      </xsd:complexContent>
   </xsd:complexType>
```

```
</xsd:element>
<xsd:element name="string_question" substitutionGroup="inter:question">
   <xsd:annotation>
      <xsd:documentation> A string_question is a type of question element that requires a
         string answer. </xsd:documentation>
   </xsd:annotation>
   <xsd:complexType>
      <xsd:complexContent>
         <xsd:extension base="inter:QuestionType">
            <xsd:attribute name="default_answer" type="xsd:string" use="optional">
               <xsd:annotation>
                  <xsd:documentation> An optional default value may be specified as the
                     answer. </xsd:documentation>
               </xsd:annotation>
            </xsd:attribute>
         </xsd:extension>
      </xsd:complexContent>
   </xsd:complexType>
</xsd:element>

<!-- ********************************************************************** -->
<!-- *   TestAction Elements                                              * -->
<!-- ********************************************************************** -->
<xsd:element name="test_action" type="inter:ItemBaseType" abstract="true">
   <xsd:annotation>
      <xsd:documentation>This is a common base element for the question_test_action element.
      </xsd:documentation>
   </xsd:annotation>
</xsd:element>
<xsd:element name="question_test_action" type="inter:QuestionTestActionType" abstract="true"
   substitutionGroup="inter:test_action">
   <xsd:annotation>
      <xsd:documentation> The question_test_action element contains a reference to a single
         question along with a set of handlers that indicate how processing should proceed
         based on the answer provided by the user. This element is abstract and is
         implemented in a document as a boolean_test_action, choice_test_action,
         numeric_test_action, or string_test_action. The type of question_test_action must
         match the type of question referenced. (E.g. a boolean_test_action MUST reference a
         boolean_question, etc..) </xsd:documentation>
   </xsd:annotation>
</xsd:element>
<xsd:element name="boolean_question_test_action"
   substitutionGroup="inter:question_test_action">
   <xsd:annotation>
      <xsd:documentation> A boolean_question_test_action element references a boolean_question
         and includes handlers for TRUE (YES) or FALSE (NO) responses. </xsd:documentation>
   </xsd:annotation>
   <xsd:complexType>
      <xsd:complexContent>
         <xsd:extension base="inter:QuestionTestActionType">
            <xsd:sequence>
               <xsd:element name="when_true" type="inter:ResultChoiceType" minOccurs="1"
                  maxOccurs="1">
                  <xsd:annotation>
                     <xsd:documentation>The element when_true specifies the action to do
                        when the answer is true.</xsd:documentation>
                  </xsd:annotation>
               </xsd:element>
               <xsd:element name="when_false" type="inter:ResultChoiceType" minOccurs="1"
                  maxOccurs="1">
                  <xsd:annotation>
                     <xsd:documentation>The element when_false specifies the action to do
                        when the answer is false.</xsd:documentation>
                  </xsd:annotation>
               </xsd:element>
            </xsd:sequence>
         </xsd:extension>
      </xsd:complexContent>
   </xsd:complexType>
</xsd:element>
<xsd:element name="choice_question_test_action"
```

```
    substitutionGroup="inter:question_test_action">
    <xsd:annotation>
       <xsd:documentation>A choice_question_test_action element references a choice_question
          and includes handlers for the various choices set out in the choice_question.
       </xsd:documentation>
    </xsd:annotation>
    <xsd:complexType>
       <xsd:complexContent>
          <xsd:extension base="inter:QuestionTestActionType">
             <xsd:sequence>
                <xsd:element ref="inter:when_choice" minOccurs="1" maxOccurs="unbounded">
                   <xsd:annotation>
                      <xsd:documentation> Specifies the action to perform when the
                         indicated choice is selected by the user. </xsd:documentation>
                   </xsd:annotation>
                </xsd:element>
             </xsd:sequence>
          </xsd:extension>
       </xsd:complexContent>
    </xsd:complexType>
    <xsd:unique name="choiceRefUniqueInTestAction">
       <xsd:annotation>
          <xsd:documentation> Ensure that no choice_test_action has multiple branches for a
             single choice </xsd:documentation>
       </xsd:annotation>
       <xsd:selector xpath="inter:when_choice/inter:choice_ref"/>
       <xsd:field xpath="."/>
    </xsd:unique>

 </xsd:element>
 <xsd:element name="numeric_question_test_action"
    substitutionGroup="inter:question_test_action">
    <xsd:annotation>
       <xsd:documentation>A numeric_question_test_action element references a numeric_question
          and includes handlers that indicate actions to perform based on whether the user's
          response matches a particular value or falls within a particular range.
       </xsd:documentation>
    </xsd:annotation>
    <xsd:complexType>
       <xsd:complexContent>
          <xsd:extension base="inter:QuestionTestActionType">
             <xsd:choice>
                <xsd:annotation>
                   <xsd:documentation> This structure is used to ensure that any number of
                      when_equals and when_range handlers may appear, but there must be at
                      least one handler (of one type or the other) and any when_equals
                      handlers must precede any when_range. </xsd:documentation>
                </xsd:annotation>
                <xsd:sequence>
                   <xsd:element ref="inter:when_equals" minOccurs="1" maxOccurs="unbounded">
                      <xsd:annotation>
                         <xsd:documentation> This element holds information on what to do
                            when the answer matches the specified value.
                         </xsd:documentation>
                      </xsd:annotation>
                   </xsd:element>
                   <xsd:element ref="inter:when_range" minOccurs="0" maxOccurs="unbounded">
                      <xsd:annotation>
                         <xsd:documentation> This element holds information on what to do
                            when the answer is within a specified range of values.
                         </xsd:documentation>
                      </xsd:annotation>
                   </xsd:element>
                </xsd:sequence>
                <xsd:element ref="inter:when_range" minOccurs="1" maxOccurs="unbounded">
                   <xsd:annotation>
                      <xsd:documentation> This element holds information on what to do
                         when the answer is within a specified range of values.
                      </xsd:documentation>
                   </xsd:annotation>
                </xsd:element>
```

```
            </xsd:choice>
         </xsd:extension>
      </xsd:complexContent>
   </xsd:complexType>
</xsd:element>
<xsd:element name="string_question_test_action"
   substitutionGroup="inter:question_test_action">
   <xsd:annotation>
      <xsd:documentation> A string_question_test_action element references a string_question
         and includes handlers that indicate actions to perform based on whether the user's
         response matches a given regular expression. </xsd:documentation>
   </xsd:annotation>
   <xsd:complexType>
      <xsd:complexContent>
         <xsd:extension base="inter:QuestionTestActionType">
            <xsd:sequence>
               <xsd:element ref="inter:when_pattern" minOccurs="1" maxOccurs="unbounded">
                  <xsd:annotation>
                     <xsd:documentation> This element holds information on what to do
                        when the answer matches a specified regular expression pattern.
                     </xsd:documentation>
                  </xsd:annotation>
               </xsd:element>
            </xsd:sequence>
         </xsd:extension>
      </xsd:complexContent>
   </xsd:complexType>
</xsd:element>

<!-- ********************************************************************** -->
<!-- *   Result Elements                                                  * -->
<!-- ********************************************************************** -->
<xsd:element name="results">
   <xsd:annotation>
      <xsd:documentation> The results element stores information about the results of
         processing the questionnaires, test_actions, and questions in a document.
      </xsd:documentation>
   </xsd:annotation>
   <xsd:complexType>
      <xsd:sequence>
         <xsd:element name="title" type="inter:TextType" minOccurs="0" maxOccurs="1">
            <xsd:annotation>
               <xsd:documentation> The title element contains a descriptive heading or
                  caption describing the result set. </xsd:documentation>
            </xsd:annotation>
         </xsd:element>
         <xsd:element name="target" type="xsd:string" minOccurs="0" maxOccurs="unbounded">
            <xsd:annotation>
               <xsd:documentation> The target element contains identifying information
                  about the host that was targeted by the assessment. </xsd:documentation>
            </xsd:annotation>
         </xsd:element>
         <xsd:element name="target_address" type="xsd:string" minOccurs="0"
            maxOccurs="unbounded">
            <xsd:annotation>
               <xsd:documentation> The target_address element contains address information
                  about the host that was targeted by the assessment. </xsd:documentation>
            </xsd:annotation>
         </xsd:element>
         <xsd:element name="questionnaire_result" minOccurs="0" maxOccurs="unbounded">
            <xsd:annotation>
               <xsd:documentation> The questionnaire_result element contains information
                  about the result of a particular questionnaire. </xsd:documentation>
            </xsd:annotation>
            <xsd:complexType>
               <xsd:attribute name="questionnaire_ref" type="inter:QuestionnaireIDPattern"
                  use="required">
                  <xsd:annotation>
                     <xsd:documentation> The questionnaire_ref attribute identifies a
                        particular questionnaire using its id. </xsd:documentation>
                  </xsd:annotation>
```

```
            </xsd:attribute>
            <xsd:attribute name="result" type="inter:ResultType" use="required">
               <xsd:annotation>
                  <xsd:documentation> The result attribute holds the result of
                     evaluating the specified questionnaire. </xsd:documentation>
               </xsd:annotation>
            </xsd:attribute>
         </xsd:complexType>
      </xsd:element>
      <xsd:element ref="inter:question_result" minOccurs="0" maxOccurs="unbounded">
         <xsd:annotation>
            <xsd:documentation> A question_result element contains result information
               associated with a specific question. One of these elements (or rather,
               one of its derrived elements) will appear for each question
            evaluated.</xsd:documentation>
         </xsd:annotation>
      </xsd:element>
      <xsd:element name="test_action_result" minOccurs="0" maxOccurs="unbounded">
         <xsd:annotation>
            <xsd:documentation> The test_action_result element contains the result of a
               test_action evaluation. One of these elements will appear for each
               test_action evaluated.</xsd:documentation>
         </xsd:annotation>
         <xsd:complexType>
            <xsd:attribute name="test_action_ref" type="inter:TestActionRefValuePattern"
               use="required">
               <xsd:annotation>
                  <xsd:documentation> The test_action_ref attribute identifies a
                     specific test_action using its id. </xsd:documentation>
               </xsd:annotation>
            </xsd:attribute>
            <xsd:attribute name="result" type="inter:ResultType" use="required">
               <xsd:annotation>
                  <xsd:documentation> The result attribute holds the result of
                     evaluating the specified test_action specified.
                  </xsd:documentation>
               </xsd:annotation>
            </xsd:attribute>
         </xsd:complexType>
      </xsd:element>
   </xsd:sequence>
   <xsd:attribute name="start_time" type="xsd:dateTime" use="optional">
      <xsd:annotation>
         <xsd:documentation> The start_time attribute is an optional attribute the
            specifies when the evaluation of this OCIL file started.
         </xsd:documentation>
      </xsd:annotation>
   </xsd:attribute>
   <xsd:attribute name="end_time" type="xsd:dateTime" use="optional">
      <xsd:annotation>
         <xsd:documentation> The end_time attribute is an optional attribute the
            specifies when the evaluation of this OCIL file ended. </xsd:documentation>
      </xsd:annotation>
   </xsd:attribute>
</xsd:complexType>
</xsd:element>
<xsd:element name="question_result" type="inter:QuestionResultType" abstract="true">
   <xsd:annotation>
      <xsd:documentation> A question_result element contains result information associated
         with a specific question. The specific type of question_result
         (boolean_question_result, choice_question_result, etc.) depends on the type of the
         associated question (boolean_question, choice_question, etc.) </xsd:documentation>
   </xsd:annotation>
</xsd:element>
<xsd:element name="boolean_question_result" substitutionGroup="inter:question_result">
   <xsd:annotation>
      <xsd:documentation> A boolean_question_result element contains a reference to a
         boolean_question, the user's response, and whether the question was successfully
         posed. </xsd:documentation>
   </xsd:annotation>
   <xsd:complexType>
```

```
      <xsd:complexContent>
         <xsd:extension base="inter:QuestionResultType">
            <xsd:sequence>
               <xsd:element name="answer" type="xsd:boolean" maxOccurs="1" nillable="true">
                  <xsd:annotation>
                     <xsd:documentation>The value of the answer to the boolean_question.
                        It could either be TRUE or FALSE.</xsd:documentation>
                  </xsd:annotation>
               </xsd:element>
            </xsd:sequence>
         </xsd:extension>
      </xsd:complexContent>
   </xsd:complexType>
</xsd:element>
<xsd:element name="choice_question_result" substitutionGroup="inter:question_result">
   <xsd:annotation>
      <xsd:documentation> A choice_question_result element contains a reference to a
         choice_question, the user's response, and whether the question was successfully
         posed. </xsd:documentation>
   </xsd:annotation>
   <xsd:complexType>
      <xsd:complexContent>
         <xsd:extension base="inter:QuestionResultType">
            <xsd:sequence>
               <xsd:element name="answer" maxOccurs="1" nillable="true">
                  <xsd:annotation>
                     <xsd:documentation>The answer element contains a choice_ref
                        attribute that identifies the choice selected by the
                        user.</xsd:documentation>
                  </xsd:annotation>
                  <xsd:complexType>
                     <xsd:attribute name="choice_ref" type="inter:ChoiceIDPattern">
                        <xsd:annotation>
                           <xsd:documentation>The choice_ref attribute specifies an id
                              of the choice selected by the user.</xsd:documentation>
                        </xsd:annotation>
                     </xsd:attribute>
                  </xsd:complexType>
               </xsd:element>
            </xsd:sequence>
         </xsd:extension>
      </xsd:complexContent>
   </xsd:complexType>
</xsd:element>
<xsd:element name="numeric_question_result" substitutionGroup="inter:question_result">
   <xsd:annotation>
      <xsd:documentation> A numeric_question_result element contains a reference to a
         numeric_question, the result provided by the user, and whether the question was
         successfully posed. </xsd:documentation>
   </xsd:annotation>
   <xsd:complexType>
      <xsd:complexContent>
         <xsd:extension base="inter:QuestionResultType">
            <xsd:sequence>
               <xsd:element name="answer" type="xsd:decimal" maxOccurs="1" nillable="true">
                  <xsd:annotation>
                     <xsd:documentation>The decimal value of the answer to a
                        numeric_question as provided by the user. </xsd:documentation>
                  </xsd:annotation>
               </xsd:element>
            </xsd:sequence>
         </xsd:extension>
      </xsd:complexContent>
   </xsd:complexType>
</xsd:element>
<xsd:element name="string_question_result" substitutionGroup="inter:question_result">
   <xsd:annotation>
      <xsd:documentation> A string_question_result element contains a reference to a
         string_question, the string provided by the user in response, and whether the
         question was successfully posed. </xsd:documentation>
   </xsd:annotation>
```

```
   <xsd:complexType>
      <xsd:complexContent>
         <xsd:extension base="inter:QuestionResultType">
            <xsd:sequence>
               <xsd:element name="answer" type="xsd:string" maxOccurs="1" nillable="true">
                  <xsd:annotation>
                     <xsd:documentation> The string value of the answer to a
                        string_question provided by the user. </xsd:documentation>
                  </xsd:annotation>
               </xsd:element>
            </xsd:sequence>
         </xsd:extension>
      </xsd:complexContent>
   </xsd:complexType>
</xsd:element>

<!-- ********************************************************************** -->
<!-- *   Global Types                                                       * -->
<!-- ********************************************************************** -->
<xsd:simpleType name="ScopeType">
   <xsd:restriction base="xsd:string">
      <xsd:enumeration value="FULL">
         <xsd:annotation>
            <xsd:documentation> The FULL value indicates that all questions must be asked
               regardless of whether or not it they are all needed to produce a result for
               a questionnaire. </xsd:documentation>
         </xsd:annotation>
      </xsd:enumeration>
      <xsd:enumeration value="SHORT">
         <xsd:annotation>
            <xsd:documentation> The SHORT value indicates that once a result value can be
               computed for a questionnaire, then it is safe to stop asking questions.
            </xsd:documentation>
         </xsd:annotation>
      </xsd:enumeration>
   </xsd:restriction>
</xsd:simpleType>
<xsd:simpleType name="BooleanQuestionModelType">
   <xsd:annotation>
      <xsd:documentation> Provides the acceptable models (i.e. set of acceptable responses)
         for a boolean_question. </xsd:documentation>
   </xsd:annotation>
   <xsd:restriction base="xsd:string">
      <xsd:enumeration value="MODEL_YES_NO">
         <xsd:annotation>
            <xsd:documentation> MODEL_YES_NO represents a response set of YES, NO.
            </xsd:documentation>
         </xsd:annotation>
      </xsd:enumeration>
      <xsd:enumeration value="MODEL_TRUE_FALSE">
         <xsd:annotation>
            <xsd:documentation> MODEL_TRUE_FALSE represents a response set of TRUE, FALSE.
            </xsd:documentation>
         </xsd:annotation>
      </xsd:enumeration>
   </xsd:restriction>
</xsd:simpleType>
<xsd:simpleType name="OperatorType">
   <xsd:restriction base="xsd:string">
      <xsd:enumeration value="AND">
         <xsd:annotation>
            <xsd:documentation> The AND operator produces a true result if every argument is
               true. If one or more arguments are false, the result of the AND is false.
               See the truth table provided in the ResultType type for a complete list of
               how the various result types are combined by an AND operation.
            </xsd:documentation>
         </xsd:annotation>
      </xsd:enumeration>
      <xsd:enumeration value="OR">
         <xsd:annotation>
            <xsd:documentation> The OR operator produces a true result if one or more
```

```
                arguments is true. If every argument is false, the result of the OR is
                false. See the truth table provided in the ResultType type for a complete
                list of how the various result types are combined by an AND
             operation.</xsd:documentation>
          </xsd:annotation>
       </xsd:enumeration>
    </xsd:restriction>
 </xsd:simpleType>
 <xsd:complexType name="CompoundTestActionType">
    <xsd:annotation>
       <xsd:documentation>The CompoundTestActionType type describes the structures used to
          combine multiple test_action elements into a single result.</xsd:documentation>
    </xsd:annotation>
    <xsd:complexContent>
       <xsd:extension base="inter:ItemBaseType">
          <xsd:sequence>
             <xsd:element name="title" type="inter:TextType" minOccurs="0" maxOccurs="1">
                <xsd:annotation>
                   <xsd:documentation> The title element contains a descriptive heading for
                      this set of test_actions. </xsd:documentation>
                </xsd:annotation>
             </xsd:element>
             <xsd:element name="description" type="inter:TextType" minOccurs="0"
                maxOccurs="1">
                <xsd:annotation>
                   <xsd:documentation> The description element contains a caption
                      describing the set of test_actions. </xsd:documentation>
                </xsd:annotation>
             </xsd:element>
             <xsd:element name="reference" type="inter:ReferenceType" minOccurs="0"
                maxOccurs="unbounded">
                <xsd:annotation>
                   <xsd:documentation> The reference element contains information about any
                      external references related to this step. Examples could include
                      references to other standards such as CVE, CCE, or CPE.
                   </xsd:documentation>
                </xsd:annotation>
             </xsd:element>
             <xsd:element name="actions" type="inter:OperationType" minOccurs="1"
                maxOccurs="1">
                <xsd:annotation>
                   <xsd:documentation> The actions element holds one or more test_action
                      elements along with the operators used to combine them into a single
                      result.</xsd:documentation>
                </xsd:annotation>
             </xsd:element>
          </xsd:sequence>
       </xsd:extension>
    </xsd:complexContent>
 </xsd:complexType>
 <xsd:complexType name="GeneratorType">
    <xsd:annotation>
       <xsd:documentation> The GeneratorType type defines an element that is used to hold
          information about when a particular OCIL document was generated, what version of the
          schema was used, what tool was used to generate the document, and what version of
          the tool was used. </xsd:documentation>
       <xsd:documentation>Additional generator information is also allowed although it is not
          part of the official OCIL language. Individual organizations can place generator
          information that they feel are important. </xsd:documentation>
    </xsd:annotation>
    <xsd:sequence>
       <xsd:element name="product_name" type="xsd:string" minOccurs="0" maxOccurs="1">
          <xsd:annotation>
             <xsd:documentation>The optional product_name specifies the name of the
                application used to generate the file.</xsd:documentation>
          </xsd:annotation>
       </xsd:element>
       <xsd:element name="product_version" type="xsd:string" minOccurs="0" maxOccurs="1">
          <xsd:annotation>
             <xsd:documentation>The optional product_version specifies the version of the
                application used to generate the file.</xsd:documentation>
```

```
            </xsd:annotation>
         </xsd:element>
         <xsd:element name="author" minOccurs="0" maxOccurs="unbounded">
            <xsd:annotation>
               <xsd:documentation> Identifies one of the authors of this document
               </xsd:documentation>
            </xsd:annotation>
            <xsd:complexType>
               <xsd:simpleContent>
                  <xsd:extension base="xsd:string">
                     <xsd:attribute name="organization" type="xsd:string" use="optional">
                        <xsd:annotation>
                           <xsd:documentation> Optionally, identify the organization for
                              whom this author works. </xsd:documentation>
                        </xsd:annotation>
                     </xsd:attribute>
                  </xsd:extension>
               </xsd:simpleContent>
            </xsd:complexType>
         </xsd:element>
         <xsd:element name="schema_version" type="xsd:decimal" minOccurs="1" maxOccurs="1">
            <xsd:annotation>
               <xsd:documentation>The required schema_version specifies the version of the OCIL
                  schema that the document has been written in and that should be used for
                  validation.</xsd:documentation>
            </xsd:annotation>
         </xsd:element>
         <xsd:element name="timestamp" type="xsd:dateTime" minOccurs="1" maxOccurs="1">
            <xsd:annotation>
               <xsd:documentation>The required timestamp specifies when the particular OCIL
                  document was generated. The format for the timestamp is yyyy-mm-ddThh:mm:ss.
               </xsd:documentation>
            </xsd:annotation>
         </xsd:element>
         <xsd:any minOccurs="0" maxOccurs="unbounded" processContents="skip"/>
      </xsd:sequence>
   </xsd:complexType>
   <xsd:complexType name="DocumentType">
      <xsd:annotation>
         <xsd:documentation> This type describes structures used to provide document-level
            information, including title, descriptions, and notices. </xsd:documentation>
      </xsd:annotation>
      <xsd:sequence>
         <xsd:element name="title" type="xsd:string" minOccurs="1" maxOccurs="1">
            <xsd:annotation>
               <xsd:documentation> Used to provide a title for this document
               </xsd:documentation>
            </xsd:annotation>
         </xsd:element>
         <xsd:element name="description" type="xsd:string" minOccurs="0" maxOccurs="unbounded">
            <xsd:annotation>
               <xsd:documentation> Each description element contains part of an overall
                  description for the entire document. (Note that questionnaires contain their
                  own description for questionnaire specific descriptions.)
               </xsd:documentation>
            </xsd:annotation>
         </xsd:element>
         <xsd:element name="notice" type="xsd:string" minOccurs="0" maxOccurs="unbounded">
            <xsd:annotation>
               <xsd:documentation> Each notice contains a notice or warning to the user of this
                  document. </xsd:documentation>
            </xsd:annotation>
         </xsd:element>
      </xsd:sequence>
   </xsd:complexType>
   <xsd:complexType name="ItemBaseType">
      <xsd:annotation>
         <xsd:documentation> The ItemBaseType complex type defines structures allowing a set of
            notes to be included. This type is inherited by many of the elements in the OCIL
            language.</xsd:documentation>
      </xsd:annotation>
```

```
   <xsd:sequence>
      <xsd:element name="notes" type="xsd:string" minOccurs="0" maxOccurs="unbounded">
         <xsd:annotation>
            <xsd:documentation> An optional set of notes to describe additional information.
            </xsd:documentation>
         </xsd:annotation>
      </xsd:element>
   </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="OperationType">
   <xsd:annotation>
      <xsd:documentation> The OperationType type defines structures that hold a set of
         test_actions and provide instructions as to how to aggregate their individual
         results into a single result. </xsd:documentation>
   </xsd:annotation>
   <xsd:sequence>
      <xsd:element ref="inter:test_action_ref" minOccurs="1" maxOccurs="unbounded">
         <xsd:annotation>
            <xsd:documentation> The test_action_ref elements holds the identifier of a
               test_action element. At least one test_action_ref must be included.
            </xsd:documentation>
         </xsd:annotation>
      </xsd:element>
   </xsd:sequence>
   <xsd:attribute name="operation" default="AND" type="inter:OperatorType">
      <xsd:annotation>
         <xsd:documentation> The operation attribute describes how to aggregate the results
            of a set of testActions. Its value defaults to the Boolean operator "AND".
         </xsd:documentation>
      </xsd:annotation>
   </xsd:attribute>
   <xsd:attribute name="negate" type="xsd:boolean" default="false">
      <xsd:annotation>
         <xsd:documentation> The negate attribute can be used to specify whether to toggle
            the result from PASS to FAIL, and vice versa. A result other than PASS or FAIL
            (e.g. ERROR, NOT_TESTED, etc.) will be unchanged by a negate operation.
         </xsd:documentation>
      </xsd:annotation>
   </xsd:attribute>
   <xsd:attribute name="priority" type="inter:PriorityType" use="optional" default="LOW">
      <xsd:annotation>
         <xsd:documentation> Priority is an optional attribute that can either be HIGH,
            MEDIUM or LOW. It specifies the importance of the referenced set of test_action
            elements. </xsd:documentation>
      </xsd:annotation>
   </xsd:attribute>
</xsd:complexType>
<xsd:complexType name="PatternType">
   <xsd:annotation>
      <xsd:documentation>The pattern element specifies a regular expression against which a
         string will be compared. </xsd:documentation>
   </xsd:annotation>
   <xsd:simpleContent>
      <xsd:extension base="xsd:string">
         <xsd:annotation>
            <xsd:documentation>This contents of this field must be a Perl Compatable Regular
               Expression (PCRE). </xsd:documentation>
         </xsd:annotation>
      </xsd:extension>
   </xsd:simpleContent>
</xsd:complexType>
<xsd:simpleType name="PriorityType">
   <xsd:annotation>
      <xsd:documentation> This type provides the possible priorities of a set of test_actions.
      </xsd:documentation>
   </xsd:annotation>
   <xsd:restriction base="xsd:string">
      <xsd:enumeration value="HIGH"/>
      <xsd:enumeration value="MEDIUM"/>
      <xsd:enumeration value="LOW"/>
   </xsd:restriction>
```

```
</xsd:simpleType>
<xsd:complexType name="QuestionTestActionType">
   <xsd:annotation>
      <xsd:documentation> The QuestionTestActionType type defines structures that are used to
         hold handlers for non-standard results (UNKNOWN, NOT_TESTED, NOT_APPLICABLE, and
         ERROR) received from a referenced question. All children of question_test_action
         extend this type. </xsd:documentation>
   </xsd:annotation>
   <xsd:complexContent>
      <xsd:extension base="inter:ItemBaseType">
         <xsd:sequence>
            <xsd:element name="title" type="inter:TextType" minOccurs="0" maxOccurs="1">
               <xsd:annotation>
                  <xsd:documentation> The title element contains a descriptive heading for
                     this set of handlers. </xsd:documentation>
               </xsd:annotation>
            </xsd:element>
            <xsd:element name="when_unknown" type="inter:ResultChoiceType" minOccurs="0">
               <xsd:annotation>
                  <xsd:documentation> The when_unknown element contains processing
                     instructions for when when the received result is UNKNOWN.
                  </xsd:documentation>
               </xsd:annotation>
            </xsd:element>
            <xsd:element name="when_not_tested" type="inter:ResultChoiceType" minOccurs="0">
               <xsd:annotation>
                  <xsd:documentation> The when_not_tested element contains processing
                     instructions for when when the received result is NOT_TESTED.
                  </xsd:documentation>
               </xsd:annotation>
            </xsd:element>
            <xsd:element name="when_not_applicable" type="inter:ResultChoiceType"
               minOccurs="0">
               <xsd:annotation>
                  <xsd:documentation> The when_not_applicable element contains processing
                     instructions for when when the received result is NOT_APPLICABLE.
                  </xsd:documentation>
               </xsd:annotation>
            </xsd:element>
            <xsd:element name="when_error" type="inter:ResultChoiceType" minOccurs="0">
               <xsd:annotation>
                  <xsd:documentation> The when_error element contains processing
                     instructions for when when the received result is ERROR.
                  </xsd:documentation>
               </xsd:annotation>
            </xsd:element>
         </xsd:sequence>
         <xsd:attribute name="question_ref" type="inter:QuestionIDPattern" use="required">
            <xsd:annotation>
               <xsd:documentation> The question_ref attribute contains the id value of a
                  question element. </xsd:documentation>
            </xsd:annotation>
         </xsd:attribute>
         <xsd:attribute name="id" type="inter:QuestionTestActionIDPattern" use="required">
            <xsd:annotation>
               <xsd:documentation> Each item is required to have a unique identifier that
                  conforms to the definition of NCName in the Recommendation "Namespaces
                  in XML 1.0", i.e., all XML 1.0 names that does not contain colons.
               </xsd:documentation>
            </xsd:annotation>
         </xsd:attribute>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>
<xsd:complexType name="QuestionResultType">
   <xsd:annotation>
      <xsd:documentation> The QuestionResultType complex type defines structures that hold
         information about a question and the user's response to it. </xsd:documentation>
   </xsd:annotation>
   <xsd:attribute name="question_ref" type="inter:QuestionIDPattern" use="required">
      <xsd:annotation>
```

```
            <xsd:documentation> The question_ref attribute contains the id of a question.
            </xsd:documentation>
         </xsd:annotation>
      </xsd:attribute>
      <xsd:attribute name="response" type="inter:UserResponseType" use="optional"
         default="ANSWERED">
         <xsd:annotation>
            <xsd:documentation>The response attribute classifies the user response. If the user
               provides a standard answer to the question the response is set to ANSWERED (the
               default). If, however, the user selects an exceptional answer (UNKNOWN,
               NOT_APPLICABLE, etc.) then this attribute will be set to the corresponding
               exceptional result. </xsd:documentation>
         </xsd:annotation>
      </xsd:attribute>
   </xsd:complexType>
   <xsd:complexType name="QuestionType">
      <xsd:annotation>
         <xsd:documentation> The QuestionType complex type defines a structure to describe a
            question and any instructions to help in determining an answer. </xsd:documentation>
      </xsd:annotation>
      <xsd:complexContent>
         <xsd:extension base="inter:ItemBaseType">
            <xsd:sequence>
               <xsd:element name="question_text" type="xsd:string" minOccurs="1"
                  maxOccurs="unbounded">
                  <xsd:annotation>
                     <xsd:documentation> Provides the text of the question to pose to the
                        user. </xsd:documentation>
                  </xsd:annotation>
               </xsd:element>
               <xsd:element ref="inter:instructions" minOccurs="0" maxOccurs="1">
                  <xsd:annotation>
                     <xsd:documentation> An optional instructions field may be included to
                        hold additional instructions to assist the user in determining the
                        answer to the question. </xsd:documentation>
                  </xsd:annotation>
               </xsd:element>
            </xsd:sequence>
            <xsd:attribute name="id" type="inter:QuestionIDPattern" use="required">
               <xsd:annotation>
                  <xsd:documentation> Each item is required to have a unique identifier that
                     conforms to the definition of NCName in the Recommendation "Namespaces
                     in XML 1.0", i.e., all XML 1.0 names that does not contain colons.
                  </xsd:documentation>
               </xsd:annotation>
            </xsd:attribute>
         </xsd:extension>
      </xsd:complexContent>
   </xsd:complexType>
   <xsd:complexType name="RangeType">
      <xsd:annotation>
         <xsd:documentation> This type describes structures to define a range against which a
            numeric user response is to be compared. </xsd:documentation>
      </xsd:annotation>
      <xsd:sequence>
         <xsd:element name="min" minOccurs="0" maxOccurs="1">
            <xsd:annotation>
               <xsd:documentation> The min element contains a minimum value.
               </xsd:documentation>
            </xsd:annotation>
            <xsd:complexType>
               <xsd:simpleContent>
                  <xsd:extension base="xsd:decimal">
                     <xsd:attribute name="inclusive" type="xsd:boolean" default="true">
                        <xsd:annotation>
                           <xsd:documentation> The inclusive attribute specifies whether
                              the minimum value should be in the specified range. The
                              default is true, indicating it is included.
                           </xsd:documentation>
                        </xsd:annotation>
                     </xsd:attribute>
```

```
                </xsd:extension>
             </xsd:simpleContent>
          </xsd:complexType>
       </xsd:element>
       <xsd:element name="max" minOccurs="0" maxOccurs="1">
          <xsd:annotation>
             <xsd:documentation> The max element contains a maximum value.
             </xsd:documentation>
          </xsd:annotation>
          <xsd:complexType>
             <xsd:simpleContent>
                <xsd:extension base="xsd:decimal">
                   <xsd:attribute name="inclusive" type="xsd:boolean" default="true">
                      <xsd:annotation>
                         <xsd:documentation> The inclusive attribute specifies whether
                            the minimum value should be included in the range. The
                            default is true, indicating it is included.
                         </xsd:documentation>
                      </xsd:annotation>
                   </xsd:attribute>
                </xsd:extension>
             </xsd:simpleContent>
          </xsd:complexType>
       </xsd:element>
    </xsd:sequence>
 </xsd:complexType>
 <xsd:complexType name="ResultChoiceType">
    <xsd:annotation>
       <xsd:documentation> The ResultChoiceType complex type specifies processing instructions
          - either produce a result or move on to another test. The ResultChoiceType is
          extended by all handlers ("when_...") in test_actions.</xsd:documentation>
    </xsd:annotation>
    <xsd:choice>
       <xsd:element name="result" type="inter:ResultType">
          <xsd:annotation>
             <xsd:documentation> This element indicates that a final value (i.e. PASS, FAIL,
                ERROR, UNKNOWN, NOT_TESTED, NOT_APPLICABLE) should be returned if the
                encapsulating handler is invoked. </xsd:documentation>
          </xsd:annotation>
       </xsd:element>
       <xsd:element ref="inter:test_action_ref">
          <xsd:annotation>
             <xsd:documentation> This element indicates that a new test_action should be
                processed if the encapsulating handler is invoked. </xsd:documentation>
          </xsd:annotation>
       </xsd:element>
    </xsd:choice>
 </xsd:complexType>
 <xsd:simpleType name="ExceptionalResultType">
    <xsd:annotation>
       <xsd:documentation> The possible exceptional results of a question </xsd:documentation>
    </xsd:annotation>
    <xsd:restriction base="xsd:string">
       <xsd:enumeration value="UNKNOWN">
          <xsd:annotation>
             <xsd:documentation>An UNKNOWN value indicates that the result of a test cannot
                be determined.</xsd:documentation>
          </xsd:annotation>
       </xsd:enumeration>
       <xsd:enumeration value="ERROR">
          <xsd:annotation>
             <xsd:documentation>An ERROR value indicates that an error occured while
                processing the check.</xsd:documentation>
             <xsd:documentation> Among other causes, this can indicate an unexpected response
                from the user. </xsd:documentation>
          </xsd:annotation>
       </xsd:enumeration>
       <xsd:enumeration value="NOT_TESTED">
          <xsd:annotation>
             <xsd:documentation>A NOT_TESTED value indicates that the check has not been
                tested yet. </xsd:documentation>
```

```
            </xsd:annotation>
         </xsd:enumeration>
         <xsd:enumeration value="NOT_APPLICABLE">
            <xsd:annotation>
               <xsd:documentation>A NOT_APPLICABLE value indicates that the check is not
                  relevant and can be skipped.</xsd:documentation>
            </xsd:annotation>
         </xsd:enumeration>
      </xsd:restriction>
   </xsd:simpleType>
   <xsd:simpleType name="ResultType">
      <xsd:annotation>
         <xsd:documentation>The ResultType simple type defines acceptable result values for
            questionnaires and test_actions.</xsd:documentation>
         <xsd:documentation> || P | F | E | U | NT | NA ||
            ---------------||------------------------------||------------------||-----------------
-------------------------
            || 1+ | 0 | 0 | 0 | 0 | 0+ || Pass || 0+ | 1+ | 0+ | 0+ | 0+ | 0+ || Fail AND || 0+
            | 0 | 1+ | 0+ | 0+ | 0+ || Error || 0+ | 0 | 0 | 1+ | 0+ | 0+ || Unknown || 0+ | 0 |
            0 | 0 | 1+ | 0+ || Not Tested || 0 | 0 | 0 | 0 | 0 | 1+ || Not Applicable || 0 | 0 |
            0 | 0 | 0 | 0 || Not Tested
            ---------------||------------------------------||------------------||-----------------
-------------------------
            || 1+ | 0+ | 0+ | 0+ | 0+ | 0+ || Pass || 0 | 1+ | 0 | 0 | 0 | 0+ || Fail OR || 0 |
            0+ | 1+ | 0+ | 0+ | 0+ || Error || 0 | 0+ | 0 | 1+ | 0+ | 0+ || Unknown || 0 | 0+ |
            0 | 0 | 1+ | 0+ || Not Tested || 0 | 0 | 0 | 0 | 0 | 1+ || Not Applicable || 0 | 0 |
            0 | 0 | 0 | 0 || Not Tested </xsd:documentation>
      </xsd:annotation>
      <xsd:union memberTypes="inter:ExceptionalResultType">
         <xsd:simpleType>
            <xsd:restriction base="xsd:string">
               <xsd:enumeration value="PASS">
                  <xsd:annotation>
                     <xsd:documentation>A PASS value indicates that the check passed its
                        test.</xsd:documentation>
                  </xsd:annotation>
               </xsd:enumeration>
               <xsd:enumeration value="FAIL">
                  <xsd:annotation>
                     <xsd:documentation>A FAIL value indicates that the check did not pass
                        its test.</xsd:documentation>
                  </xsd:annotation>
               </xsd:enumeration>
            </xsd:restriction>
         </xsd:simpleType>
      </xsd:union>
   </xsd:simpleType>
   <xsd:simpleType name="UserResponseType">
      <xsd:union memberTypes="inter:ExceptionalResultType">
         <xsd:simpleType>
            <xsd:restriction base="xsd:string">
               <xsd:enumeration value="ANSWERED">
                  <xsd:annotation>
                     <xsd:documentation>Indicates that the user responded by answering the
                        question posed</xsd:documentation>
                  </xsd:annotation>
               </xsd:enumeration>
            </xsd:restriction>
         </xsd:simpleType>
      </xsd:union>
   </xsd:simpleType>
   <xsd:complexType name="TextType">
      <xsd:annotation>
         <xsd:documentation> The TextType complex type defines an element that holds any
            information. </xsd:documentation>
      </xsd:annotation>
      <xsd:complexContent>
         <xsd:extension base="xsd:anyType">
            <xsd:attribute ref="xml:lang">
               <xsd:annotation>
                  <xsd:documentation> This attribute specifies the language in which to
```

```
                interpret the information. </xsd:documentation>
            </xsd:annotation>
         </xsd:attribute>
      </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>

<!-- ********************************************************************** -->
<!-- *   ID Patterns                                                      * -->
<!-- ********************************************************************** -->
<xsd:simpleType name="QuestionnaireIDPattern">
   <xsd:annotation>
      <xsd:documentation> ID values for questionnaires must match this pattern.
      </xsd:documentation>
   </xsd:annotation>
   <xsd:restriction base="xsd:string">
      <xsd:pattern value="ocil:[A-Za-z0-9_]+:questionnaire:[1-9][0-9]*"/>
   </xsd:restriction>
</xsd:simpleType>
<xsd:simpleType name="QuestionIDPattern">
   <xsd:annotation>
      <xsd:documentation> ID values for questions must match this pattern. Each ID must be
         unique within an OCIL document. </xsd:documentation>
   </xsd:annotation>
   <xsd:restriction base="xsd:string">
      <xsd:pattern value="ocil:[A-Za-z0-9_]+:question:[1-9][0-9]*"/>
   </xsd:restriction>
</xsd:simpleType>
<xsd:simpleType name="QuestionTestActionIDPattern">
   <xsd:annotation>
      <xsd:documentation> ID values for test_actions must match this pattern. Each ID must be
         unique within an OCIL document. </xsd:documentation>
   </xsd:annotation>
   <xsd:restriction base="xsd:string">
      <xsd:pattern value="ocil:[A-Za-z0-9_]+:testaction:[1-9][0-9]*"/>
   </xsd:restriction>
</xsd:simpleType>
<xsd:simpleType name="TestActionRefValuePattern">
   <xsd:annotation>
      <xsd:documentation> A test_action_ref may refer to either a test_action or a
         questionnaire. This type represents the union of these two ID patterns.
      </xsd:documentation>
   </xsd:annotation>
   <xsd:restriction base="xsd:string">
      <xsd:pattern value="ocil:[A-Za-z0-9_]+:testaction:[1-9][0-9]*"/>
      <xsd:pattern value="ocil:[A-Za-z0-9_]+:questionnaire:[1-9][0-9]*"/>
   </xsd:restriction>
</xsd:simpleType>
<xsd:simpleType name="ChoiceIDPattern">
   <xsd:annotation>
      <xsd:documentation> ID values for choices in choice_questions must match this pattern.
         Each ID must be unique within an OCIL document. </xsd:documentation>
   </xsd:annotation>
   <xsd:restriction base="xsd:string">
      <xsd:pattern value="ocil:[A-Za-z0-9_]+:choice:[1-9][0-9]*"/>
   </xsd:restriction>
</xsd:simpleType>
<xsd:simpleType name="ChoiceGroupIDPattern">
   <xsd:annotation>
      <xsd:documentation> ID values for choice_group references in choice_questions must match
         this pattern. Each ID must be unique within an OCIL document. </xsd:documentation>
   </xsd:annotation>
   <xsd:restriction base="xsd:string">
      <xsd:pattern value="ocil:[A-Za-z0-9_]+:choicegroup:[1-9][0-9]*"/>
   </xsd:restriction>
</xsd:simpleType>

<!-- ********************************************************************** -->
<!-- *   Global Elements                                                  * -->
<!-- ********************************************************************** -->
<xsd:element name="test_action_ref">
```

```
   <xsd:annotation>
      <xsd:documentation> The test_action_ref element holds a reference (id) to a test_action
         or questionnaire. </xsd:documentation>
   </xsd:annotation>
   <xsd:complexType>
      <xsd:simpleContent>
         <xsd:extension base="inter:TestActionRefValuePattern">
            <xsd:attribute name="negate" type="xsd:boolean" default="false">
               <xsd:annotation>
                  <xsd:documentation> The negate attribute can be used to specify whether
                     to toggle the result from PASS to FAIL, and vice versa. A result
                     other than PASS or FAIL (e.g. ERROR, NOT_TESTED, etc.) will be
                     unchanged by a negate operation. </xsd:documentation>
               </xsd:annotation>
            </xsd:attribute>
            <xsd:attribute name="priority" type="inter:PriorityType" use="optional"
               default="LOW">
               <xsd:annotation>
                  <xsd:documentation> Priority is an optional attribute that can either be
                     HIGH, MEDIUM or LOW. It specifies the importance of a priority test
                     action reference. </xsd:documentation>
               </xsd:annotation>
            </xsd:attribute>
         </xsd:extension>
      </xsd:simpleContent>
   </xsd:complexType>
</xsd:element>
<xsd:element name="when_choice">
   <xsd:annotation>
      <xsd:documentation>The element when_choice specifies the action to take in a
         choice_test_action when a particular choice is selected by a user in response to a
         choice_question. </xsd:documentation>
   </xsd:annotation>
   <xsd:complexType>
      <xsd:complexContent>
         <xsd:extension base="inter:ResultChoiceType">
            <xsd:sequence>
               <xsd:element name="choice_ref" type="inter:ChoiceIDPattern"
                  maxOccurs="unbounded">
                  <xsd:annotation>
                     <xsd:documentation>The choice_ref element specifies the id of a
                        choice. </xsd:documentation>
                  </xsd:annotation>
               </xsd:element>
            </xsd:sequence>
         </xsd:extension>
      </xsd:complexContent>
   </xsd:complexType>
</xsd:element>
<xsd:element name="choice">
   <xsd:annotation>
      <xsd:documentation> A choice element holds information about one acceptable answer to a
         choice_question. </xsd:documentation>
   </xsd:annotation>
   <xsd:complexType>
      <xsd:simpleContent>
         <xsd:extension base="xsd:string">
            <xsd:attribute name="id" type="inter:ChoiceIDPattern" use="required">
               <xsd:annotation>
                  <xsd:documentation> All choices are tagged with a unique identifier that
                     may be referenced by a choice_test_action referencing the
                     encapsulating choice_question. </xsd:documentation>
               </xsd:annotation>
            </xsd:attribute>
         </xsd:extension>
      </xsd:simpleContent>
   </xsd:complexType>
</xsd:element>
<xsd:element name="choice_group">
   <xsd:annotation>
      <xsd:documentation> A choice_group defines a group of choices that may then be reused in
```

```
         multiple choice_question elements. For example, a document may include multiple
         choice_questions with the options of "Good", "Fair", or "Poor". By defining these
         choices in a single choice_group, the author would not need to list them out
         explicitly in every choice_question. </xsd:documentation>
   </xsd:annotation>
   <xsd:complexType>
      <xsd:sequence>
         <xsd:element ref="inter:choice" minOccurs="1" maxOccurs="unbounded">
            <xsd:annotation>
               <xsd:documentation> Holds the information associated with one of the
                  possible responses for a choice_question. </xsd:documentation>
            </xsd:annotation>
         </xsd:element>
      </xsd:sequence>
      <xsd:attribute name="id" type="inter:ChoiceGroupIDPattern" use="required">
         <xsd:annotation>
            <xsd:documentation> Holds the id of this choice group. This id is referenced
               within choice_question elements to include the choices contained in a group.
            </xsd:documentation>
         </xsd:annotation>
      </xsd:attribute>
   </xsd:complexType>
</xsd:element>
<xsd:element name="when_equals">
   <xsd:annotation>
      <xsd:documentation> The element when_equals specifies the action to take in a
         numeric_test_action when a particular value is given by a user in response to a
         numeric_question. </xsd:documentation>
   </xsd:annotation>
   <xsd:complexType>
      <xsd:complexContent>
         <xsd:extension base="inter:ResultChoiceType">
            <xsd:sequence>
               <xsd:element name="value" type="xsd:decimal" minOccurs="1"
                  maxOccurs="unbounded">
                  <xsd:annotation>
                     <xsd:documentation> Each value holds what is to be matched.
                     </xsd:documentation>
                  </xsd:annotation>
               </xsd:element>
            </xsd:sequence>
         </xsd:extension>
      </xsd:complexContent>
   </xsd:complexType>
</xsd:element>
<xsd:element name="when_range">
   <xsd:annotation>
      <xsd:documentation> The element when_range specifies the action to take in a
         numeric_test_action when a value given by a user in response to a numeric_question
         falls within the indicated range. </xsd:documentation>
   </xsd:annotation>
   <xsd:complexType>
      <xsd:complexContent>
         <xsd:extension base="inter:ResultChoiceType">
            <xsd:sequence>
               <xsd:element name="range" type="inter:RangeType" minOccurs="1"
                  maxOccurs="unbounded">
                  <xsd:annotation>
                     <xsd:documentation> Each range element holds a single numeric range.
                     </xsd:documentation>
                  </xsd:annotation>
               </xsd:element>
            </xsd:sequence>
         </xsd:extension>
      </xsd:complexContent>
   </xsd:complexType>
</xsd:element>
<xsd:element name="when_pattern">
   <xsd:annotation>
      <xsd:documentation> The element when_pattern specifies the action to take in a
         string_test_action when a string given by a user in response to a string_question
```

```
         matches the given regular expression. </xsd:documentation>
      </xsd:annotation>
      <xsd:complexType>
         <xsd:complexContent>
            <xsd:extension base="inter:ResultChoiceType">
               <xsd:sequence>
                  <xsd:element name="pattern" type="inter:PatternType" minOccurs="1"
                     maxOccurs="unbounded">
                     <xsd:annotation>
                        <xsd:documentation> Each pattern element holds a regular expression
                           against which the user's response string is to be compared.
                        </xsd:documentation>
                     </xsd:annotation>
                  </xsd:element>
               </xsd:sequence>
            </xsd:extension>
         </xsd:complexContent>
      </xsd:complexType>
   </xsd:element>


   <!-- ********************************************************************** -->
   <!-- *   Instructions Element and Supporting Types                        * -->
   <!-- ********************************************************************** -->
   <xsd:complexType name="ReferenceType" mixed="true">
      <xsd:annotation>
         <xsd:documentation> The ReferenceType complex type defines structures used to hold
            information about an external reference given its URI and description.
         </xsd:documentation>
         <xsd:documentation> This structure may be used to reference other standards such as CVE,
            CCE, or CPE. To do so, the href attribute would give the relevant namespace. For
            example, the namespace of the current version of CPE is
            http://cpe.mitre.org/dictionary/2.0 and the body of this element would hold a
            specific CPE identifier. References to other information (documents, web pages,
            etc.) are also permitted. </xsd:documentation>
      </xsd:annotation>
      <xsd:complexContent>
         <xsd:extension base="inter:TextType">
            <xsd:attribute name="href" type="xsd:anyURI">
               <xsd:annotation>
                  <xsd:documentation> The href attribute holds the URI of an external
                     reference. This may be the namespace associated with the information in
                     the body or a web URL containing relevant information.
                  </xsd:documentation>
               </xsd:annotation>
            </xsd:attribute>
         </xsd:extension>
      </xsd:complexContent>
   </xsd:complexType>
   <xsd:complexType name="StepType">
      <xsd:annotation>
         <xsd:documentation> The StepType complex type defines structures that describe one step
            (out of possibly multiple steps) that a user should take in order to respond to a
            question. The steps would appear as parts of the question's instructions element.
         </xsd:documentation>
      </xsd:annotation>
      <xsd:sequence>
         <xsd:element name="description" type="inter:TextType" minOccurs="0" maxOccurs="1">
            <xsd:annotation>
               <xsd:documentation> The description element contains some information about this
                  step. </xsd:documentation>
            </xsd:annotation>
         </xsd:element>
         <xsd:element name="reference" minOccurs="0" maxOccurs="unbounded"
            type="inter:ReferenceType">
            <xsd:annotation>
               <xsd:documentation> The reference element contains information about any
                  external references related to this step. </xsd:documentation>
            </xsd:annotation>
         </xsd:element>
         <xsd:element ref="inter:step" minOccurs="0" maxOccurs="unbounded">
```

```
            <xsd:annotation>
               <xsd:documentation> The step element contains substeps for this particular step.
               </xsd:documentation>
            </xsd:annotation>
         </xsd:element>
      </xsd:sequence>
      <xsd:attribute name="is_done" type="xsd:boolean" default="false">
         <xsd:annotation>
            <xsd:documentation> The is_done attribute indicates whether this step has been done.
               The value is true when it is done. Otherwise, it is false, It is an optional
               attribute that defaults to false. </xsd:documentation>
         </xsd:annotation>
      </xsd:attribute>
      <xsd:attribute name="is_required" type="xsd:boolean" default="true">
         <xsd:annotation>
            <xsd:documentation>The is_required attribute indicates whether a step is required or
               not. If it is not, then it can be skipped. It is an optional attribute that
               defaults to true.</xsd:documentation>
         </xsd:annotation>
      </xsd:attribute>
   </xsd:complexType>
   <xsd:element name="step" type="inter:StepType">
      <xsd:annotation>
         <xsd:documentation>The step element describes one step in the procedures a user should
            undertake in order to answer an encapsulating question.</xsd:documentation>
      </xsd:annotation>
   </xsd:element>
   <xsd:element name="instructions">
      <xsd:annotation>
         <xsd:documentation> The instructions element contains a step by step procedure to guide
            the user in answering a question. </xsd:documentation>
      </xsd:annotation>
      <xsd:complexType>
         <xsd:sequence>
            <xsd:element name="title" type="inter:TextType">
               <xsd:annotation>
                  <xsd:documentation> The title element contains a descriptive heading for the
                     instructions. </xsd:documentation>
               </xsd:annotation>
            </xsd:element>
            <xsd:element ref="inter:step" minOccurs="1" maxOccurs="unbounded">
               <xsd:annotation>
                  <xsd:documentation> The step element contains information about one step of
                     the instructions. </xsd:documentation>
               </xsd:annotation>
            </xsd:element>
         </xsd:sequence>
      </xsd:complexType>
   </xsd:element>
</xsd:schema>
```