Machine Authentication of MRTDs for Public Sector Applications

Part 3: High Level Document Check Interface Specification

Version 2.5
# Table of Contents

1. Changelog ................................................................................................................. 1
   1.1. Changelog 2.5 Part 3 ......................................................................................... 1

2. Introduction .................................................................................................................. 2
   2.1. Motivation ........................................................................................................... 2

3. Architecture for Inspection Applications .................................................................... 3
   3.1. Client-Server Architecture ................................................................................ 3
   3.2. Workflow-Based Document Check ......................................................................... 3
   3.3. Transaction Logging .............................................................................................. 4
   3.4. Cross-document combined check ........................................................................... 4

4. Document overview ...................................................................................................... 6
   4.1. Terminology ......................................................................................................... 6
   4.2. Naming conventions ............................................................................................. 6
   4.3. Namespaces .......................................................................................................... 7
   4.4. XML Schema and Web Service Definition ........................................................... 7
   4.5. Interoperability ..................................................................................................... 7

5. Interface overview ........................................................................................................ 8
   5.1. High-Level Document Check ............................................................................... 8
   5.2. Transaction management interface ....................................................................... 9
   5.3. Error handling ...................................................................................................... 9

6. High Level Document Check API .............................................................................. 10
   6.1. Namespaces ....................................................................................................... 10
   6.2. Data types ........................................................................................................... 10
   6.3. Fault types .......................................................................................................... 14
   6.4. Operations ........................................................................................................... 16
   6.5. Workflow definition schema ............................................................................... 26

7. Workflow feedback schema ......................................................................................... 62
   7.1. Feedback document ............................................................................................. 62
   7.2. type.feedback ....................................................................................................... 62
   7.3. type.feedback.dg1 ............................................................................................... 64
   7.4. type.feedback.dg2 ............................................................................................... 66
   7.5. type.feedback.dg2.template ................................................................................. 66
   7.6. type.feedback.dg2.image .................................................................................... 67
   7.7. type.feedback.dg3 ............................................................................................... 68
   7.8. type.feedback.dg3.template ................................................................................. 68
   7.9. type.feedback.dg3.image .................................................................................... 68
   7.10. type.feedback.dg3.image.fingerpos ................................................................. 69
   7.11. type.feedback.dg4 ............................................................................................. 70
7.12. type.feedback.dg4.template ................................................................. 70
7.13. type.feedback.dg4.image ......................................................................... 71
7.14. type.feedback.dg11 .................................................................................. 71
7.15. type.feedback.dg12 .................................................................................. 73
7.16. type.feedback.optmrz .............................................................................. 74
7.17. type.feedback.eid.placeofresidence .......................................................... 75
7.18. type.feedback.eid.placeofresidence.structuredplace .................................. 76
7.19. type.feedback.defects .............................................................................. 76
7.20. type.feedback.oid .................................................................................... 77
7.21. type.feedback.defects.defect ................................................................... 77
7.22. type.feedbacks.name .............................................................................. 78
7.23. type.feedbacks.name .............................................................................. 78
7.24. type.feedback.checkresult ...................................................................... 79
7.25. type.feedback.xpathresult .................................................................... 79
7.26. type.feedback.seal.visa .......................................................................... 79
7.27. type.feedback.seal.visa.durationofstay ..................................................... 80
7.28. type.feedback.seal.emergencytraveldocument ........................................ 81
7.29. type.feedback.seal.arrivalattestationdocument ....................................... 81
7.30. type.feedback.seal.socialinsurancecard ................................................... 82
7.31. type.feedback.seal.residencepermit ........................................................ 82
7.32. type.feedback.seal.residencepermitsupplementarysheet" .......................... 83
7.33. type.feedback.seal.addressstickergermanidentitycard" ............................. 83
7.34. type.feedback.seal.unknown .................................................................. 84
7.35. type.feedback.mrz.unfolded ................................................................... 84
7.36. type.feedback.mrz.unfolded.shortened .................................................... 86
7.37. type.feedback.string.date ...................................................................... 87
7.38. type.feedback.extended .......................................................................... 88
7.39. Workflow extensions ................................................................................ 88
8.  Transaction management .............................................................................. 90
8.1. Namespace ................................................................................................ 90
8.2. Data types .................................................................................................. 90
8.3. Fault types .................................................................................................. 91
8.4. Operations ................................................................................................ 93

List of Abbreviations ......................................................................................... 105
Bibliography ....................................................................................................... 106
List of Figures

3.1. Client-side document check process ................................................................. 3
3.2. Extension of with -compliant logging .................................................................. 4
3.3. Extension of accounting for cross-document combined checks. .......................... 5
1. Changelog

The following tables present the changes introduced between the latest versions of this Technical Guideline. The changelog lists the changes grouped per part of this Technical Guideline, per element (section, table, figure) and type of change, refer to [KeepAChangelog]:

- Added for new features
- Changed for changes in existing functionality
- Deprecated for soon-to-be removed features
- Removed for now removed features
- Fixed for any bug fixes
- Security in case of vulnerabilities

1.1. Changelog 2.5 Part 3

<table>
<thead>
<tr>
<th>Element Name</th>
<th>Type of Change</th>
<th>Change Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chapter High Level Document Check API</td>
<td>Added</td>
<td>New workflow elements:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• AggregatedCheck</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• SealCheck</td>
</tr>
<tr>
<td>Chapter Workflow feedback schema</td>
<td>Added</td>
<td>New feedback elements:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• type.feedback.seal.visa</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• type.feedback.seal.visa.durationofstay</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• type.feedback.seal.emergencytraveldocument</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• type.feedback.seal.arrivalattestationdocument</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• type.feedback.seal.socialinsurancecard</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• type.feedback.seal.socialinsurancecard.socialinsurancenumber</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• type.feedback.seal.residencepermit</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• type.feedback.seal.residencepermitsupplementarysheet</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• type.feedback.seal.addressstickergermanidentitycard</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• type.feedback.seal.unknown</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• type.feedback.mrz.unfolded</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• type.feedback.mrz.unfolded.shortened</td>
</tr>
</tbody>
</table>

|Table 1.1 Changelog BSI TR-03135, Part 3|
2. Introduction

This technical guideline specifies two complimentary web services that provide validation of Machine-Readable Travel Documents (MRTDs) according to [BSI TR-03135-1]. They comprise the document check process as well as the required logging of the results.

2.1. Motivation

The checking of Machine Readable Travel Documents (MRTDs) according to [BSI TR-03135-1] requires a large number of individual steps. Some steps require a certain order of execution or may only be relevant for particular documents. In contrast to a fixed definition of scenario-specific check processes, the processing and visualization of results and/or document data may vary with the purpose of the Inspection Application. The complexity of the document check process causes large efforts for the creation and quality control of Inspection Applications.

The goal of this document is to provide high-level interfaces that reduces the programming effort for Inspection Applications by separating the document check process and standard-compliant logging from the problem-specific processing and visualization of results.
3. Architecture for Inspection Applications

3.1. Client-Server Architecture

To ease the implementation of Inspection Applications for document checks, they are split into two parts. Frontends, or clients, are application-specific. They provide a user interface and process and visualize results depending on their particular purpose. The [BSI TR-03135]-compliant document check process and the corresponding transaction logging are provided by a backend server as web services. This separation reduces the effort to implement Inspection Applications for different purposes considerably.

3.2. Workflow-Based Document Check

To further reduce the effort for client implementation, the document check process is configured based on a textual description in Extensible Markup Language (XML) rather than program code. A particular configuration is referred to as workflow, its description as workflow definition. This definition allows the customization of the application-scenario-specific document check process within the limits of [BSI TR-03135].

It controls which optical, electronic and combined checks are carried out and which data are required for processing. The document data and check results are referred to as feedback. Please refer to Section 6.5 for an exhaustive specification of the workflow definition and Chapter 7 for the specification of the feedback data format.

The workflow definition can be provided by the client or the server. This allows clients to use custom workflows but also allows centralized management on the server. All workflows are managed by the server and are available to all clients. Workflows are executed on the server upon request by the client. Relevant feedback is generated on the server and can be fetched by the client individually.

![Figure 3.1. Client-side document check process](image)

Figure 3.1 visualizes the client side of the document check process. Workflow-related interface operations are highlighted in orange. After a new document is detected, the client requests execution of the relevant workflow. In the following, the client individually fetches the feedback elements that are defined in the workflow from the server and processes them appropriately, e.g. by visualizing them to the user. After all feedback has been processed successfully, the client informs the server that client-side processing has finished. Section 5.1 provides a quick overview of the interface functions. The interface specification can be found in Chapter 6.
3.3. Transaction Logging

[BSI TR-03135-1] requires logging of all relevant check results which is provided by the interface specified in Chapter 8. If multiple documents need to be checked, the corresponding transactions can be merged into a single transaction for consistent logging of related document checks.

Figure 3.2. Extension of Figure 3.1 with [BSI TR-03135] - compliant logging

» Figure 3.2 shows the extension of the client side of the document check process from » Figure 3.1 with transaction logging operations (highlighted in blue, dashed). The client initiates a new transaction after a MRTD was detected and before initiating workflow execution. After the workflow execution and result processing is finished, the client requests saving of the corresponding transaction log and then ends the transaction. » Section 5.2 provides a quick overview of the interface functions.

3.4. Cross-document combined check

Part 1 of this Technical Guideline specifies checks across two different documents (e.g. passport and visa) or two sides of the same document (e.g. front and rear side of ID cards).

Initially, both documents (or document sides) are checked individually. Cross-document combined checks, which are always defined for a specific document (e.g. visa), are skipped.

Its the responsibility of the client to control which documents are considered for cross-document combined checks. After linking the respective documents, the cross-document combined checks are executed.

The Inspection Application MUST merge the corresponding transaction to ensure consistent logging of the checking process.

» Figure 3.3 shows the extension of the single document checking process in » Figure 3.2 which accounts for potential cross-document combined checks. After finishing the initial checking of a document, the client checks if feedback elements for any cross-document combined checks are missing. If so, the client needs to link the currently checked document (e.g. visa) to the relevant reference document (e.g. passport) and re-request the corresponding feedback elements. Afterwards, the corresponding transactions for both documents must be merged.
Figure 3.3. Extension of Figure 3.2 accounting for cross-document combined checks.
4. Document overview

4.1. Terminology
The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

4.2. Naming conventions

4.2.1. Multiplicity
Generally, XML elements and attributes listed in this document are required, i.e. the respective parent element MUST contain exactly one such element. Elements and attributes that deviate from this baseline are denoted in this document by a symbol which is appended to the element/attribute name. The symbols are listed in Table 4.1.

<table>
<thead>
<tr>
<th>Appended symbol</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>?</td>
<td>Zero or one</td>
</tr>
<tr>
<td>*</td>
<td>Zero or more</td>
</tr>
<tr>
<td>+</td>
<td>One or more</td>
</tr>
</tbody>
</table>

Table 4.1 Multiplicity symbols

4.2.2. SOAP Interfaces
All operations of this interface follow the request/response model, i.e., communication is initiated by the client by sending a Simple Object Access Protocol (SOAP) message to the server (request). For each request, the server replies with a SOAP message containing the result of the requested operation (response) or, in case of error, a fault.

The body of each SOAP message consists of a single part which is named according to the corresponding operation. For requests, the part name is identical to the name of the operation. For responses, the part name is identical to the name of the operation plus the suffix "Response" (see Table 4.2).

<table>
<thead>
<tr>
<th>Message type</th>
<th>Part name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Request</td>
<td>&lt;operation_name&gt;</td>
</tr>
<tr>
<td>Response</td>
<td>&lt;operation_name&gt;Response</td>
</tr>
</tbody>
</table>

Table 4.2 Naming convention for SOAP messages

Example: Naming convention
- Operation: getAllWorkflows
- Request: getAllWorkflows
- Response: getAllWorkflowsResponse

Both request and response elements exclusively contain zero or more child elements according to the detailed description in this guideline. They do not carry any attributes.
4.3. Namespaces

The namespaces used in the schemata are given in Table 4.3.

<table>
<thead>
<tr>
<th>Prefix</th>
<th>Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>hldc</td>
<td>High Level Document Check</td>
<td><a href="http://trdoccheck.bsi.bund.de/hl-dc/wsd1/2">http://trdoccheck.bsi.bund.de/hl-dc/wsd1/2</a></td>
</tr>
<tr>
<td>wf</td>
<td>High Level Document Check (HLDC)</td>
<td><a href="http://trdoccheck.bsi.bund.de/hl-dc/workflow/2">http://trdoccheck.bsi.bund.de/hl-dc/workflow/2</a></td>
</tr>
<tr>
<td>tl</td>
<td>Transaction management</td>
<td><a href="http://trdoccheck.bsi.bund.de/tl/">http://trdoccheck.bsi.bund.de/tl/</a> wsdl/2</td>
</tr>
<tr>
<td>xs</td>
<td>XML Schema</td>
<td><a href="http://www.w3.org/2001/XMLSchema">http://www.w3.org/2001/XMLSchema</a></td>
</tr>
</tbody>
</table>

Table 4.3 Namespaces

4.4. XML Schema and Web Service Definition

The XML Schema Definition (XSD) (.xsd) and Web Services Description Language (WSDL) (.wsdl) files given in Table 4.4 are provided with this Technical Guideline.

<table>
<thead>
<tr>
<th>Message type</th>
<th>Part name</th>
</tr>
</thead>
<tbody>
<tr>
<td>hldc_v2.wsdl</td>
<td>HLDC web service definition (Chapter 6)</td>
</tr>
<tr>
<td>hldc_v2.xsd</td>
<td>XSD for HLDC web service</td>
</tr>
<tr>
<td>wf_v2.xsd</td>
<td>XSD for workflow definitions (Section 6.5)</td>
</tr>
<tr>
<td>fb_v2.xsd</td>
<td>XSD for workflow feedback (Chapter 7)</td>
</tr>
<tr>
<td>tl_v2.wsdl</td>
<td>Transaction management web service definition (Chapter 8)</td>
</tr>
<tr>
<td>tl_v2.xsd</td>
<td>XSD for transaction management web service</td>
</tr>
</tbody>
</table>

Table 4.4 Naming convention for SOAP messages

4.5. Interoperability

To ensure trouble-free interoperability between different SOAP implementations, both client and server implementations SHOULD fulfill the WS-I Basic Profile 1.1.
5. Interface overview

5.1. High-Level Document Check

5.1.1. Objective
HLDC interface provides execution of [BSI TR-03135]-compliant document checks controlled by XML-based workflow definitions.

5.1.2. Document detection
The client MAY choose to be notified of a newly detected document by explicit waiting (blocking).
• Calling `waitForNewDocument` blocks client execution until a new document is detected (limited by a timeout).

5.1.3. Workflow management
• The client MAY publish custom workflow definitions on the server by calling `addWorkflow`. Identically named workflow definitions are replaced.
• The client MAY remove workflow definitions from the server by calling `removeWorkflow`.
• The function `getAllWorkflows` provides a list of all workflow definitions that are available on the server.

5.1.4. Workflow execution
• The function `beginWorkflow` executes a document check workflow. If no transaction ID is provided, the server MUST initiate a new transaction internally to ensure [BSI TR-03135]-compliant logging. `beginWorkflow` provides a unique workflow ID that MUST be used in subsequent calls. In general, the ID MUST remain valid until the client calls `endWorkflow`. To protect against broken clients, the server MAY limit the ID validity with a timeout.
• Workflow execution MAY be cancelled by calling `cancelWorkflow`.
• The client MUST inform the server that client-side workflow processing has finished by calling `endWorkflow`.

5.1.5. Workflow feedback
Document and check result data are provided by the server individually and are referred to as workflow feedback.
• The function `getWorkflowFeedback` provides the next available feedback element.
• The client MAY request particular document and result data by calling `getWorkflowFeedbackById`.

5.1.6. Cross-document checks
If a workflow contains definitions for cross-document combined checks, the client SHOULD call `linkWorkflow` in order to assign the relevant reference document. The call MUST appear after feedback processing for the current document has finished.
5.2. Transaction management interface

5.2.1. Objective

The transaction management interface provides functions to link document check operations to a [BSI TR-03135]-compliant transaction and to allow logging of relevant data.

5.2.2. Transaction management

- Transactions are initiated by `beginTransaction`. In contrast to calling the HLDC function `beginWorkflow` without a transaction ID, this function provides an ID that is REQUIRED for all the functions of this interface and MAY be used when calling `beginWorkflow` in order to link the workflow execution to this particular transaction.
- Multiple transactions MAY be merged into a single one with `mergeTransaction`.
- Transactions are terminated by `endTransaction` which invalidates the transaction ID.

5.2.3. Logging

- The function `saveTransaction` stores the currently available logged transaction data persistently on the server. The client MAY limit the amount of data that is stored.
- The client MAY request the currently available transaction data for client-side use by calling `getTransactionXML`.
- Client-specific data MAY be logged by calling `addLogData`.
- The function `getAllLoggingProviders` is OPTIONAL and MAY return a server-specific list of logging targets that MAY be used when calling `saveTransaction`.
- The function `saveTransactionXML` allows to store client-generated XML via available logging providers.
- The client MAY publish custom logging profiles (Extensible Stylesheet Language Transformation (XSLT)) on the server by calling `addLoggingProfile`. Identically named logging profiles are replaced.
- The client MAY remove logging profiles from the server by calling `removeLoggingProfile`.

5.2.4. Write-protection

In order to prevent inconsistent log data, a transaction is protected against further modification after the first read access, i.e., after calls to `saveTransaction` and `getTransactionXML`. All future write access, i.e., calls that would modify the transaction, MUST fail then with an appropriate error.

The server MAY fail read access calls with an appropriate error if the transaction is currently modified (e.g., by the document check process).

5.3. Error handling

If errors occur during processing of a web service request, a SOAP fault is generated according to the SOAP 1.1 specification. SOAP faults are comparable to exceptions in programming languages such as C++, C# or Java insofar as they allow reporting of errors without the need to account for error codes in function signatures.

SOAP faults are returned in place of the SOAP response. Depending on the type of an error, the fault message may contain additional information about the error. The faults that are specific to the web services in this document are specified in the respective chapters and listed with every function that may generate them. Faults originating from other causes such as network connection problems or validation errors are beyond the scope of this document as they depend on the specific SOAP implementation.
6. High Level Document Check API

The HLDC Application Programming Interface (API) contains functions to perform electronic and optical document checks in conformity to part 1 of this Technical Guideline. The check processes are driven by XML workflows and provide a very high level interface to the application. The check results contain multiple sub-results which are combined to overall results according to [BSI TR-03135-1]. All checks can be logged in XML format compliant to [BSI TR-03135-1] using the transaction management API in Chapter 8.

The definitions of the HLDC API are provided in hldc_v2.wsdl. The schemata for the workflow definition and XML-formatted feedback are provided in wf_v2.xsd and fb_v2.xsd respectively.

6.1. Namespaces

The elements of the server- and client-side APIs are member of the namespace http://trdoccheck.bsi.bund.de/hldc/wsdl/2, which is aliased by hldc. The workflow definition schema and the workflow feedback schema use the namespace http://trdoccheck.bsi.bund.de/hldc/workflow/2 aliased by wf.

6.2. Data types

In addition to simple XSD types, the SOAP interface uses custom data types, which are described in the following.

6.2.1. FeedbackStatus

Represents the status of a feedback element. Derived from xs:string.

6.2.1.1. Values

The values of the FeedbackStatus values are given in Table 6.1.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>valid</td>
<td>The feedback element contains valid data which is available in string-Feedback or binaryFeedback.</td>
</tr>
<tr>
<td>err-invalid-conversion</td>
<td>An error occurred during feedback preparation. Data could not be converted into the specified target format.</td>
</tr>
<tr>
<td>err-not-available</td>
<td>Requested data is not available or could not be read.</td>
</tr>
<tr>
<td>err-reference-document-required</td>
<td>Requested data require access to a reference document for cross-document checks (see linkWorkflow).</td>
</tr>
</tbody>
</table>

Table 6.1 FeedbackStatus - values

6.2.1.2. WSDL Definition

```xml
<simpleType name="FeedbackStatus">
  <restriction base="xs:string"/>
  <enumeration value="valid"/>
  <enumeration value="err-invalid-conversion"/>
  <enumeration value="err-not-available"/>
</restriction>
</simpleType>
```
6.2.2. UUID

Serves to uniquely reference various elements at runtime (e.g. currently executing workflows). Inherits xs:string.

6.2.2.1. Format restrictions

The content MUST be empty or represent a universally unique identifier of 32 lower-case hexadecimal letters that are separated into 5 groups of length 8, 4, 4, 4 and 12 letters using hyphens (e.g. 01234567-89ab-cde-f-0123-456789abcdef).

6.2.2.2. WSDL Definition

```xml
<complexType name="Workflow">
  <sequence>
    <element name="name" type="xs:string"/>
    <element name="vendor" type="xs:string"/>
    <element name="version" type="xs:string"/>
    <element name="description" type="xs:string"/>
  </sequence>
</complexType>
```

6.2.3. Workflow

Contains general workflow information.

### 6.2.3.1. Elements

The Workflow elements are given in Table 6.2.

<table>
<thead>
<tr>
<th>Element name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>xs:string</td>
</tr>
<tr>
<td>vendor</td>
<td>xs:string</td>
</tr>
<tr>
<td>version</td>
<td>xs:string</td>
</tr>
<tr>
<td>description</td>
<td>xs:string</td>
</tr>
</tbody>
</table>

The name of the workflow.

The vendor that defined the workflow.

The version of the workflow.

The description of the workflow.

Table 6.2 Workflow - elements

6.2.3.2. WSDL Definition

```xml
<complexType name="Workflow">
  <sequence>
    <element name="name" type="xs:string" /></element>
    <element name="vendor" type="xs:string" />
    <element name="version" type="xs:string" />
    <element name="description" type="xs:string" />
  </sequence>
</complexType>
```

6.2.4. WorkflowFeedback

Represents a feedback element from a HLDC workflow.

### 6.2.4.1. Attributes

None.
6.2.4.2. Elements

The WorkflowFeedback elements are given in Table 6.3.

<table>
<thead>
<tr>
<th>Element name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>feedbackID</td>
<td>xs:string Reference to a feedback element. MUST be empty if status is either finished or cancelled. MUST NOT be empty otherwise.</td>
</tr>
<tr>
<td>stringFeedback?</td>
<td>xs:string The feedback data as string (if possible). MUST be empty if status is different from valid.</td>
</tr>
<tr>
<td>binaryFeedback?</td>
<td>xs:base64Binary The feedback data as binary (if possible). MUST be empty if status is different from valid.</td>
</tr>
<tr>
<td>status</td>
<td>hldc:FeedbackStatus The status code of the feedback element.</td>
</tr>
</tbody>
</table>

Table 6.3 WorkflowFeedback - elements

6.2.4.3. WSDL Definition

```xml
<complexType name="WorkflowFeedback">
  <sequence>
    <element name="feedbackID" type="xs:string" />
    <element name="stringFeedback" type="xs:string" minOccurs="0" />
    <element name="binaryFeedback" type="xs:base64Binary" minOccurs="0" maxOccurs="1" />
    <element name="status" type="hldc:FeedbackStatus" />
  </sequence>
</complexType>
```

6.2.5. WorkflowParameter

Key-value pair to configure conditions of type type.workflow.conditions.parameter in a HLDC workflow definition.

6.2.5.1. Attributes

None.

6.2.5.2. Elements

The WorkflowParameter elements are given in Table 6.4.

<table>
<thead>
<tr>
<th>Element name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>xs:string The key.</td>
</tr>
<tr>
<td>value</td>
<td>xs:boolean Specifies whether the corresponding condition MUST be considered fulfilled.</td>
</tr>
</tbody>
</table>

Table 6.4 WorkflowParameter - elements
6.2.5.3. WSDL Definition

```xml
<complexType name="WorkflowParameter">
  <sequence>
    <element name="id" type="xs:string" />
    <element name="value" type="xs:boolean" />
  </sequence>
</complexType>
```

6.2.6. WorkflowParameters

List of key-value pairs.

6.2.6.1. Elements

The WorkflowParameters elements are given in Table 6.5.

<table>
<thead>
<tr>
<th>Element name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>parameter</td>
<td>hldc:Parameter</td>
</tr>
<tr>
<td></td>
<td>A key-value pair.</td>
</tr>
</tbody>
</table>

Table 6.5 WorkflowParameters - elements

6.2.6.2. WSDL Definition

```xml
<complexType name="WorkflowParameters">
  <sequence>
    <element name="parameter" type="hldc:WorkflowParameter"
              minOccurs="0" maxOccurs="unbounded"/>
  </sequence>
</complexType>
```

6.2.7. WorkflowStatus

Represents the execution status of a workflow. Derived from xs:string.

6.2.7.1. Values

The WorkflowParameters values are given in Table 6.6.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ok</td>
<td>Workflow execution is still in progress.</td>
</tr>
<tr>
<td>finished</td>
<td>The workflow execution has finished. This is the last feedback element delivered by getWorkflowFeedback.</td>
</tr>
<tr>
<td>cancelled</td>
<td>The workflow has been cancelled by the user. This is the last feedback element delivered by getWorkflowFeedback.</td>
</tr>
</tbody>
</table>

Table 6.6 WorkflowStatus - values

6.2.7.2. WSDL Definition

```xml
<simpleType name="WorkflowStatus">
  <restriction base="xs:string">
    <enumeration value="ok"/>
    <enumeration value="finished"/>
    <enumeration value="cancelled"/>
  </restriction>
</simpleType>
```
6.3. Fault types
This section specifies the SOAP faults that are specific to this SOAP API. No fault has any attributes.

6.3.1. InvalidClientId
Returned if a client ID does not exist on the server. The ID is either invalid or has expired.

6.3.1. Elements
None

6.3.1.2. WSDL Definition

```xml
<complexType name="InvalidClientId">
  <sequence>
  </sequence>
</complexType>
```

6.3.2. InvalidConditionId
Returned if a condition ID is not defined in the workflow definition.

6.3.2.1. Elements
The InvalidConditionId elements are given in Table 6.7.

<table>
<thead>
<tr>
<th>Element name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>xs:string</td>
</tr>
<tr>
<td></td>
<td>The rejected ID.</td>
</tr>
</tbody>
</table>

Table 6.7 InvalidConditionId - elements

6.3.2.2. WSDL Definition

```xml
<complexType name="InvalidConditionId">
  <sequence>
    <element name="id" type="xs:string" />
  </sequence>
</complexType>
```

6.3.3. InvalidFeedbackId
Returned if the feedback ID in the request is not defined in the workflow definition.

6.3.3.1. Elements
None.

6.3.3.2. WSDL Definition

```xml
<complexType name="InvalidFeedbackId">
  <sequence>
  </sequence>
</complexType>
```

6.3.4. InvalidWorkflowId
Returned if a workflow ID does not exist on the server. The ID is either invalid or has expired due to a call to endWorkflow or limited resources on the server.
6.3.4.1. Elements

The InvalidWorkflowId elements are given in Table 6.7.

<table>
<thead>
<tr>
<th>Element name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id?</td>
<td>hldc:UUID</td>
</tr>
<tr>
<td></td>
<td>The rejected ID. Only present in case of ambiguity.</td>
</tr>
</tbody>
</table>

Table 6.8 InvalidWorkflowId - elements

6.3.4.2. WSDL Definition

```xml
<complexType name="InvalidWorkflowId">
  <sequence>
    <element name="id" type="hldc:UUID" minOccurs="0" maxOccurs="1" />  
  </sequence>
</complexType>
```

6.3.5. WorkflowLimitExceeded

Returned in case of too many executing workflows. The client either needs to cancel a currently executing workflow or wait until it finishes.

6.3.5.1. Elements

None.

6.3.5.2. WSDL Definition

```xml
<complexType name="WorkflowLimitExceeded">
  <sequence>
  </sequence>
</complexType>
```

6.3.6. WorkflowNotFound

Returned if the requested workflow does not exist on the server.

6.3.6.1. Elements

None.

6.3.6.2. WSDL Definition

```xml
<complexType name="WorkflowNotFound">
  <sequence>
  </sequence>
</complexType>
```

6.3.7. WorkflowParserError

Returned if the provided workflow definition could not be parsed by the server.

6.3.7.1. Elements

None.

6.3.7.2. WSDL Definition

```xml
<complexType name="WorkflowParserError">
  <sequence>
  </sequence>
</complexType>
```
6.4. Operations

6.4.1. addWorkflow

Transfers a new workflow definition to the server. Input MUST be validated against the schema \textit{wf\_v2.xsd} (\textsection{}6.5) and MUST be checked for consistency. The workflow MUST NOT contain invalid ID references and MUST NOT contain cyclic dependencies. If an identically named workflow exists on the server, it MUST be replaced with the newly submitted definitions.

6.4.1.1. Request elements

The addWorkflow request elements are given in Table 6.9.

<table>
<thead>
<tr>
<th>Element name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>workflowDefinition</td>
<td>xs:base64Binary</td>
</tr>
<tr>
<td></td>
<td>The base64-encoded XML data of the workflow definition. The XML structure is defined in \textsection{}6.5.</td>
</tr>
</tbody>
</table>

Table 6.9 addWorkflow - request elements

6.4.1.2. Response elements

The addWorkflow response elements are given in Table 6.10.

<table>
<thead>
<tr>
<th>Element name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>workflowName</td>
<td>xs:string</td>
</tr>
<tr>
<td></td>
<td>The name of the workflow which is parsed from the workflow definition.</td>
</tr>
</tbody>
</table>

Table 6.10 addWorkflow - response elements

6.4.1.3. Faults

The addWorkflow faults are given in Table 6.11.

<table>
<thead>
<tr>
<th>Fault</th>
<th>Cause</th>
</tr>
</thead>
<tbody>
<tr>
<td>hldc:WorkflowParserError</td>
<td>xs:string</td>
</tr>
<tr>
<td></td>
<td>An error occurred while parsing or validating the submitted workflow.</td>
</tr>
</tbody>
</table>

Table 6.11 addWorkflow - faults

6.4.1.4. WSDL Definition

```xml
<element name="addWorkflow">
  <complexType>
    <sequence>
      <element name="workflowDefinition" type="xs:base64Binary" />
    </sequence>
  </complexType>
</element>
```

```xml
<element name="addWorkflowResponse">
  <complexType>
    <sequence>
      <element name="workflowName" type="xs:string" />
    </sequence>
  </complexType>
</element>
```
6.4.2. beginWorkflow

Initiates workflow execution on the server to check the current document and/or read out the requested data. Check results and data MUST be queried individually by calling getWorkflowFeedback or getWorkflowFeedback-ById.

6.4.2.1. Request elements

The beginWorkflow request elements are given in Table 6.12.

<table>
<thead>
<tr>
<th>Element name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>workflowName</td>
<td>xs:string The name of the workflow to execute. MUST be a valid workflow name as returned by addWorkflow or getAllWorkflows.</td>
</tr>
<tr>
<td>transactionID?</td>
<td>tl:UUID Transaction ID returned by tl:beginTransaction (see Section 8.4.3) to link the workflow to the specified [BSI TR-03135] transaction. If omitted or empty, the server MUST generate a new ID for internal use.</td>
</tr>
<tr>
<td>docIdentifier?</td>
<td>xs:string Machine Readable Zone (MRZ) or Card Access Number (CAN) to be used to access the document instead of the automatically retrieved (if available) MRZ/CAN.</td>
</tr>
<tr>
<td>parameters?</td>
<td>hldc:WorkflowParameters List of key-value pairs which override the default values of the hldc:ParameterCondition entries in the workflow definition. The IDs MUST match the workflow definition.</td>
</tr>
</tbody>
</table>

Table 6.12 beginWorkflow - request elements

6.4.2.2. Response elements

The beginWorkflow response elements are given in Table 6.13.

<table>
<thead>
<tr>
<th>Element name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>workflowID</td>
<td>hldc:UUID Unique ID of the workflow execution which MUST be used as a reference in corresponding function calls.</td>
</tr>
</tbody>
</table>

Table 6.13 beginWorkflow - response elements

6.4.2.3. Faults

The beginWorkflow faults are given in Table 6.14.

<table>
<thead>
<tr>
<th>Fault</th>
<th>Cause</th>
</tr>
</thead>
<tbody>
<tr>
<td>hldc:InvalidConditionId</td>
<td>parameters contains at least on condition that does not match the workflow definition.</td>
</tr>
<tr>
<td>hldc:WorkflowLimitExceeded</td>
<td>Too many workflows are currently executing on the server.</td>
</tr>
<tr>
<td>hldc:WorkflowNotFound</td>
<td>The workflow workflowName does not exist on the server.</td>
</tr>
</tbody>
</table>

Table 6.14 beginWorkflow - faults
### 6.4.2.4. WSDL Definition

```xml
<!-- operation request element -->
<element name="beginWorkflow">
  <complexType>
    <sequence>
      <element name="workflowName" type="xs:string" />
      <element name="transactionID" type="tl:UUID" minOccurs="0" />
      <element name="docIdentifier" type="xs:string" minOccurs="0" />
      <element name="parameters" type="hldc:WorkflowParameters" minOccurs="0" />
    </sequence>
  </complexType>
</element>

<!-- operation response element -->
<element name="beginWorkflowResponse">
  <complexType>
    <sequence>
      <element name="workflowID" type="hldc:UUID" />
    </sequence>
  </complexType>
</element>
```

### 6.4.3. cancelWorkflow

Cancels workflow execution. The workflow is cancelled successfully when the status cancelled is returned in the feedback loop.

#### 6.4.3.1. Request elements

The cancelWorkflow request elements are given in Table 6.15.

<table>
<thead>
<tr>
<th>Element name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>workflowID</td>
<td>hldc:UUID</td>
</tr>
<tr>
<td></td>
<td>The ID of the workflow to cancel that was returned by beginWorkflow.</td>
</tr>
</tbody>
</table>

Table 6.15 cancelWorkflow - request elements

#### 6.4.3.2. Response elements

None.

#### 6.4.3.3. Faults

The cancelWorkflow faults are given in Table 6.16.

<table>
<thead>
<tr>
<th>Fault</th>
<th>Cause</th>
</tr>
</thead>
<tbody>
<tr>
<td>hldc:InvalidWorkflowId</td>
<td>Value of workflowID is invalid or has expired.</td>
</tr>
</tbody>
</table>

Table 6.16 cancelWorkflow - faults

#### 6.4.3.4. WSDL Definition

```xml
<!-- operation request element -->
<element name="cancelWorkflow">
  <complexType>
    <sequence>
      <element name="workflowID" type="hldc:UUID" />
    </sequence>
  </complexType>
</element>
```
6.4.4. endWorkflow

Informs the server that client-side workflow processing has finished. Invalidates the workflow ID. MUST be called by the client for each executed workflow. To protect against broken clients, the server MAY invalidate workflow IDs based on a timeout. SHOULD NOT be called concurrently to other functions that operate on the same workflow ID except cancelWorkflow.

6.4.4.1. Request elements

The endWorkflow request elements are given in Table 6.17.

<table>
<thead>
<tr>
<th>Element name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>workflowID</td>
<td>hldc:UUID</td>
</tr>
<tr>
<td></td>
<td>Reference to the workflow to end. If the workflow is still executing on the server, the workflow is cancelled by the server.</td>
</tr>
</tbody>
</table>

Table 6.17 endWorkflow - request elements

6.4.4.2. Response elements

The endWorkflow response elements are given in Table 6.18.

<table>
<thead>
<tr>
<th>Element name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>workflowStatus</td>
<td>hldc:WorkflowStatus</td>
</tr>
<tr>
<td></td>
<td>Final execution status of the workflow. MUST be either finished or cancelled.</td>
</tr>
</tbody>
</table>

Table 6.18 endWorkflow - response elements

6.4.4.3. Faults

The endWorkflow faults are given in Table 6.19.

<table>
<thead>
<tr>
<th>Fault</th>
<th>Cause</th>
</tr>
</thead>
<tbody>
<tr>
<td>hldc:InvalidWorkflowId</td>
<td>Value of workflowID is invalid or has expired.</td>
</tr>
</tbody>
</table>

Table 6.19 endWorkflow - faults

6.4.4.4. WSDL Definition

```xml
<sequence>
  <element name="workflowID" type="hldc:UUID" />
</sequence>
</complexType>
```
6.4.5. getAllWorkflows

Returns a list of all available workflows.

6.4.5.1. Request elements

None.

6.4.5.2. Response elements

The getAllWorkflows response elements are given in Table 6.20.

<table>
<thead>
<tr>
<th>Element name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>workflow*</td>
<td>hldc:Workflow</td>
</tr>
<tr>
<td></td>
<td>The list of all available workflows. getAllWorkflows returns one workflow element per available workflow.</td>
</tr>
</tbody>
</table>

Table 6.20 getAllWorkflows - response elements

6.4.5.3. Faults

None.

6.4.5.4. WSDL Definition

6.4.6. getWorkflowFeedback

Returns the next available feedback element during workflow execution. Generally called in a loop until finished or cancelled is returned as feedback status. The order in which individual feedback elements are returned may be different from the order in the workflow definition. This function MAY skip feedback elements that were previously explicitly requested by getWorkflowFeedbackById.
6.4.6.1. Request elements

The getWorkflowFeedback request elements are given in Table 6.21.

<table>
<thead>
<tr>
<th>Element name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>workflowID</td>
<td>hldc:UUID</td>
</tr>
<tr>
<td></td>
<td>Reference to the workflow for which the next feedback element is requested.</td>
</tr>
<tr>
<td>timeout-ms</td>
<td>xs:int</td>
</tr>
<tr>
<td></td>
<td>Timeout in milliseconds for getting the next feedback element. If no new feedback element becomes available during this time, an error is returned. The server MAY cap the value. Negative values are interpreted as “infinity” which is subject to a server-imposed limit.</td>
</tr>
</tbody>
</table>

Table 6.21 getWorkflowFeedback - request elements

6.4.6.2. Response elements

The getWorkflowFeedback response elements are given in Table 6.22.

<table>
<thead>
<tr>
<th>Element name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>workflowStatus</td>
<td>hldc:WorkflowStatus</td>
</tr>
<tr>
<td></td>
<td>Current execution status of the workflow.</td>
</tr>
<tr>
<td>timeout-expired</td>
<td>xs:boolean</td>
</tr>
<tr>
<td></td>
<td>MUST be true if no new feedback was available before the timeout expired. MUST be false if new feedback was available in time.</td>
</tr>
<tr>
<td>feedback?</td>
<td>hldc:WorkflowFeedback</td>
</tr>
<tr>
<td></td>
<td>The next available feedback element. Omitted in case of timeout.</td>
</tr>
</tbody>
</table>

Table 6.22 getWorkflowFeedback - response elements

6.4.6.3. Faults

The getWorkflowFeedback response elements are given in Table 6.23.

<table>
<thead>
<tr>
<th>Fault type</th>
<th>Cause</th>
</tr>
</thead>
<tbody>
<tr>
<td>hldc:InvalidWorkflowId</td>
<td>Value of workflowID is invalid or has expired.</td>
</tr>
</tbody>
</table>

Table 6.23 getWorkflowFeedback - faults

6.4.6.4. WSDL Definition

```xml
<!-- operation request element -->
<element name="getWorkflowFeedback">
  <complexType>
    <sequence>
      <element name="workflowID" type="hldc:UUID" />
      <element name="timeout-ms" type="xs:int" />  
    </sequence>
  </complexType>
<!-- operation response element -->
<element name="getWorkflowFeedbackResponse">
  <complexType>
    <sequence>
      <element name="workflowStatus" type="hldc:WorkflowStatus" />
      <element name="timeout-expired" type="xs:boolean" />
    </sequence>
  </complexType>
</element>
```

Federal Office for Information Security 21
6.4.7. `getWorkflowFeedbackById`

Returns the requested feedback element.

### 6.4.7.1. Request elements

The `getWorkflowFeedbackById` request elements are given in Table 6.24.

<table>
<thead>
<tr>
<th>Element name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>workflowID</td>
<td>hldc:UUID</td>
</tr>
<tr>
<td>feedbackId</td>
<td>xs:string</td>
</tr>
<tr>
<td>timeout-rms</td>
<td>hldc:int</td>
</tr>
</tbody>
</table>

Reference to the workflow for which the next feedback element is requested. The requested feedback ID. Timeout in milliseconds for getting the feedback element. If the feedback element does not become available during this time, an error is returned. The server MAY cap the value. Negative values are interpreted as "infinity" which is subject to a server-imposed limit.

Table 6.24 `getWorkflowFeedbackById` - request elements

### 6.4.7.2. Response elements

The `getWorkflowFeedbackById` response elements are given in Table 6.25.

<table>
<thead>
<tr>
<th>Element name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>workflowStatus</td>
<td>hldc:WorkflowStatus</td>
</tr>
<tr>
<td>timeout-expired</td>
<td>xs:boolean</td>
</tr>
<tr>
<td>feedback?</td>
<td>hldc:WorkflowFeedback</td>
</tr>
</tbody>
</table>

Current execution status of the workflow. MUST be `true` if the requested feedback was not available before the timeout expired. MUST be `false` if the requested feedback was available in time. The next available feedback element. Omitted in case of timeout.

Table 6.25 `getWorkflowFeedbackById` - response elements

### 6.4.7.3. Faults

The `getWorkflowFeedbackById` response elements are given in Table 6.26.

<table>
<thead>
<tr>
<th>Fault type</th>
<th>Cause</th>
</tr>
</thead>
<tbody>
<tr>
<td>hldc:InvalidFeedbackId</td>
<td>Value of <code>feedbackId</code> is not defined in the workflow definition.</td>
</tr>
<tr>
<td>hldc:InvalidWorkflowId</td>
<td>Value of <code>workflowID</code> is invalid or has expired.</td>
</tr>
</tbody>
</table>

Table 6.26 `getWorkflowFeedbackById` - faults
6.4.7.4. WSDL Definition

```xml
<!-- operation request element -->
<element name="getWorkflowFeedbackById">
  <complexType>
    <sequence>
      <element name="workflowID" type="hldc:UUID"/>
      <element name="feedbackId" type="xs:string"/>
      <element name="timeout-ms" type="xs:int"/>
    </sequence>
  </complexType>
</element>

<!-- operation response element -->
<element name="getWorkflowFeedbackByIdResponse">
  <complexType>
    <sequence>
      <element name="workflowStatus" type="hldc:WorkflowStatus"/>
      <element name="timeout-expired" type="xs:boolean"/>
      <element name="feedback" type="hldc:WorkflowFeedback" minOccurs="0"/>
    </sequence>
  </complexType>
</element>
```

6.4.8. linkWorkflow

Links two executed workflows in order to allow execution of combined checks that refer to multiple documents or document sides. If the workflow status of the workflow `checkWorkflowID` is ok or finished, the workflow status is set to ok and all configured combined checks are executed. Combined checks that require information from a different document check will fetch this information from workflow `referredWorkflowID`. If the workflow status of the workflow `checkWorkflowID` is cancelled, calling `linkWorkflow` has no effect. If `checkWorkflowID` and `referredWorkflowID` are part of different check transactions, it is RECOMMENDED to call `tl:merge-Transaction` (see ➔Section 8.4.7) for these transactions.

### 6.4.8.1. Request elements

The `linkWorkflow` request elements are given in ➔Table 6.27.

<table>
<thead>
<tr>
<th>Element name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>checkWorkflowID</td>
<td>hldc:UUID</td>
</tr>
<tr>
<td></td>
<td>Reference to the workflow that contains combined checks which refer to multiple documents or multiple sides of a document.</td>
</tr>
<tr>
<td>referredWorkflowID</td>
<td>hldc:UUID</td>
</tr>
<tr>
<td></td>
<td>Reference to the workflow from which additional data is fetched.</td>
</tr>
</tbody>
</table>

Table 6.27 linkWorkflow - request elements

### 6.4.8.2. Response elements

None.

### 6.4.8.3. Faults

The `linkWorkflow` faults are given in ➔Table 6.28.

<table>
<thead>
<tr>
<th>Fault type</th>
<th>Cause</th>
</tr>
</thead>
<tbody>
<tr>
<td>hldc:InvalidWorkflowId</td>
<td>Value of checkWorkflowID or referredWorkflowID is invalid or has expired or values of checkWorkflowID are referredWorkflowID identical.</td>
</tr>
</tbody>
</table>

Federal Office for Information Security
The transaction used by `checkWorkflowID` is write-protected and must not be modified.

<table>
<thead>
<tr>
<th>Fault type</th>
<th>Cause</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>tl:ReadOnly</code></td>
<td>The transaction used by <code>checkWorkflowID</code> is write-protected and must not be modified.</td>
</tr>
</tbody>
</table>

### 6.4.8.4. WSDL Definition

```xml
<!-- operation request element -->
<element name="linkWorkflow">
  <complexType>
    <sequence>
      <element name="checkWorkflowID" type="hldc:UUID" />
      <element name="referredWorkflowID" type="hldc:UUID" />
    </sequence>
  </complexType>
</element>

<!-- operation response element -->
<element name="linkWorkflowResponse">
  <complexType>
    <sequence>
    </sequence>
  </complexType>
</element>
```

### 6.4.9. `removeWorkflow`

Removes a workflow definition which was previously loaded with `addWorkflow`. The server MUST ensure that workflow removal does not affect the execution of currently running workflows.

#### 6.4.9.1. Request elements

The `removeWorkflow` faults are given in Table 6.29.

<table>
<thead>
<tr>
<th>Element name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>workflowName</code></td>
<td>xs:string</td>
</tr>
<tr>
<td></td>
<td>The name of the workflow to remove.</td>
</tr>
</tbody>
</table>

Table 6.29 `removeWorkflow` - request elements

#### 6.4.9.2. Response elements

None.

#### 6.4.9.3. Faults

The `removeWorkflow` faults are given in Table 6.30.

<table>
<thead>
<tr>
<th>Fault type</th>
<th>Cause</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>hldc:WorkflowNotFound</code></td>
<td>The workflow <code>workflowName</code> does not exist on the server.</td>
</tr>
</tbody>
</table>

Table 6.30 `removeWorkflow` - faults

#### 6.4.9.4. WSDL Definition

```xml
<!-- operation request element -->
<element name="removeWorkflow">
  <complexType>
    <sequence>
      <element name="workflowName" type="xs:string" />
    </sequence>
  </complexType>
</element>
```
6.4.10. waitForNewDocument

Waits for a new document to be put onto the reader. If a new document is put onto the reader between to successive calls of waitForNewDocument with the same client ID within a certain time-frame (subject to server implementation), the second of these calls to waitForNewDocument SHOULD return immediately.

6.4.10.1. Request elements

The waitForNewDocument request elements are given in Table 6.31.

<table>
<thead>
<tr>
<th>Element name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>client-id?</td>
<td>xs:string</td>
</tr>
<tr>
<td>timeout-ms</td>
<td>xs:int</td>
</tr>
</tbody>
</table>

Table 6.31 waitForNewDocument - request elements

6.4.10.2. Response elements

The waitForNewDocument response elements are given in Table 6.32.

<table>
<thead>
<tr>
<th>Element name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>client-id</td>
<td>hldc:UUID</td>
</tr>
<tr>
<td>timeout-expired</td>
<td>xs:boolean</td>
</tr>
</tbody>
</table>

Table 6.32 waitForNewDocument - response elements

6.4.10.3. WSDL Definition

```xml
<!-- operation request element -->
<element name="waitForNewDocument">
  <complexType>
    <sequence>
      <element name="client-id" type="hldc:UUID" minOccurs="0" />
    </sequence>
  </complexType>
</element>
```
6.5. Workflow definition schema

This section defines all elements of a workflow definition.

6.5.1. Workflow document

XML document that configures the execution workflow of the document check according to TR#03135 Part 1.

6.5.1.1. Root element

The Workflow root element is given in Table 6.33.

<table>
<thead>
<tr>
<th>Element name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Workflow</td>
<td>wf:type.workflow</td>
</tr>
<tr>
<td></td>
<td>Root element of the workflow definition.</td>
</tr>
</tbody>
</table>

Table 6.33 Workflow - root element

6.5.1.2. XSD Definition

```xml
<xs:element name="Workflow" type="wf:type.workflow" />
```

6.5.2. type.workflow

Root element of a HLDC workflow definition.

6.5.2.1. Attributes

The type.workflow attributes are given in Table 6.34.

<table>
<thead>
<tr>
<th>Attribute name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>schemaVersion</td>
<td>xs:decimal</td>
</tr>
<tr>
<td></td>
<td>The schemaVersion currently has the value 1.</td>
</tr>
</tbody>
</table>

Table 6.34 type.workflow - attributes

6.5.2.2. Elements

The type.workflow elements are given in Table 6.35.

<table>
<thead>
<tr>
<th>Element name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Information</td>
<td>wf:type.workflow.information</td>
</tr>
<tr>
<td></td>
<td>Basic information about the workflow.</td>
</tr>
<tr>
<td>Element name</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ApplicationScenario</td>
<td>wf:type.workflow.scenario</td>
</tr>
<tr>
<td></td>
<td>Information about the application scenario of the workflow.</td>
</tr>
<tr>
<td>RequestedSchemaVersion?</td>
<td>wf:type.workflow.versions</td>
</tr>
<tr>
<td></td>
<td>The requested schema version of the [BSI TR-03135] XML.</td>
</tr>
<tr>
<td>Conditions?</td>
<td>wf:type.workflow.conditions</td>
</tr>
<tr>
<td></td>
<td>Conditions for certain workflow actions.</td>
</tr>
<tr>
<td>AggregatedCheck?</td>
<td>wf:type.workflow.aggregation</td>
</tr>
<tr>
<td></td>
<td>Defines an overall check result that aggregates the results of optical,</td>
</tr>
<tr>
<td></td>
<td>electronic and seal checks.</td>
</tr>
<tr>
<td>ElectronicCheck?</td>
<td>wf:type.workflow.electronic</td>
</tr>
<tr>
<td></td>
<td>Defines which electronic data shall be read and when the electronic</td>
</tr>
<tr>
<td></td>
<td>checks should be performed.</td>
</tr>
<tr>
<td>OpticalCheck?</td>
<td>wf:type.workflow.optical</td>
</tr>
<tr>
<td></td>
<td>Defines which optical data shall be read and when the optical checks</td>
</tr>
<tr>
<td></td>
<td>should be performed.</td>
</tr>
<tr>
<td>SealCheck?</td>
<td>wf:type.workflow.seal</td>
</tr>
<tr>
<td></td>
<td>Defines which barcode data shall be read and when the seal checks should</td>
</tr>
<tr>
<td></td>
<td>be performed.</td>
</tr>
<tr>
<td>CombinedCheck?</td>
<td>wf:type.workflow.combined</td>
</tr>
<tr>
<td></td>
<td>Defines which combined checks shall be performed.</td>
</tr>
<tr>
<td>Feedback?</td>
<td>wf:type.workflow.feedback</td>
</tr>
<tr>
<td></td>
<td>Defines in which format the read data should be delivered to the</td>
</tr>
<tr>
<td></td>
<td>application.</td>
</tr>
<tr>
<td>Dependencies?</td>
<td>wf:type.workflow.dependencies</td>
</tr>
<tr>
<td></td>
<td>Defines dependencies between workflow actions.</td>
</tr>
<tr>
<td>Extension?</td>
<td>wf:type.workflow.extension</td>
</tr>
<tr>
<td></td>
<td>Root node for implementation-specific workflow extensions.</td>
</tr>
</tbody>
</table>

Table 6.35 type.workflow - elements

### 6.5.2.3. XSD Definition

```xml
<xs:complexType name="type.workflow">
  <xs:sequence>
    <xs:element name="Information" type="wf:type.workflow.information" />
    <xs:element name="ApplicationScenario" type="wf:type.workflow.scenario" />  
    <xs:element name="RequestedSchemaVersions" minOccurs="0" type="wf:type.workflow.versions" />  
    <xs:element name="Conditions" minOccurs="0" type="wf:type.workflow.conditions" />  
    <xs:element name="AggregatedCheck" minOccurs="0" type="wf:type.workflow.aggregation" />  
    <xs:element name="ElectronicCheck" minOccurs="0" type="wf:type.workflow.electronic" />  
    <xs:element name="OpticalCheck" minOccurs="0" type="wf:type.workflow.optical" />  
    <xs:element name="SealCheck" minOccurs="0" type="wf:type.workflow.seal" />  
    <xs:element name="CombinedCheck" minOccurs="0" type="wf:type.workflow.combined" />
  </xs:sequence>
</xs:complexType>
```
6.5.3. **type.workflow.information**

Basic information about the workflow. MUST provide a workflow name and MAY contain additional information about the vendor, version and a textual description of the workflow.

### 6.5.3.1. Attributes

None.

### 6.5.3.2. Elements

The **type.workflow** elements are given in Table 6.36.

<table>
<thead>
<tr>
<th>Element name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vendor?</td>
<td>xs:string</td>
</tr>
<tr>
<td>Name</td>
<td>xs:string</td>
</tr>
<tr>
<td>Version?</td>
<td>xs:string</td>
</tr>
<tr>
<td>Description?</td>
<td>xs:string</td>
</tr>
</tbody>
</table>

Table 6.36 type.workflow.information - elements

### 6.5.3.3. XSD Definition

```
<xs:complexType name="type.workflow.information">
  <xs:sequence>
    <xs:element name="Vendor" minOccurs="0" type="xs:string" />
    <xs:element name="Name" >
      <xs:simpleType>
        <xs:restriction base="xs:string">
          <xs:minLength value="1" />
        </xs:restriction>
      </xs:simpleType>
    </xs:element>
    <xs:element name="Version" minOccurs="0" type="xs:string" />
    <xs:element name="Description" minOccurs="0" type="xs:string" />
  </xs:sequence>
</xs:complexType>
```

### 6.5.4. **type.workflow.scenario**

Selection of the server-defined application scenario which configures which checks are executed for a given document. The scenario MAY be customized with respect to optical, electronic and combined checks.
6.5.4.1. Attributes

The type.workflow.scenario attributes are given in Table 6.37.

<table>
<thead>
<tr>
<th>Attribute name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>preset</td>
<td>xs:string</td>
</tr>
<tr>
<td></td>
<td>Selects a configuration preset. The available presets are server-defined.</td>
</tr>
</tbody>
</table>

Table 6.37 type.workflow.scenario - attributes

6.5.4.2. Elements

The type.workflow.scenario elements are given in Table 6.38.

<table>
<thead>
<tr>
<th>Element name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpticalChecks</td>
<td>Configuration of optical checks.</td>
</tr>
<tr>
<td>ElectronicChecks</td>
<td>Configuration of electronic checks.</td>
</tr>
<tr>
<td>CombinedChecks?</td>
<td>Configuration of single-document combined checks.</td>
</tr>
<tr>
<td>CrossDocumentChecks</td>
<td>Configuration of cross-document combined checks.</td>
</tr>
<tr>
<td>SealChecks</td>
<td>Configuration of seal checks.</td>
</tr>
<tr>
<td>AggregatedChecks</td>
<td>Configuration of the overall result aggregation.</td>
</tr>
</tbody>
</table>

Table 6.38 type.workflow.scenario - elements

6.5.4.3. XSD Definition

```xml
<xs:complexType name="type.workflow.scenario">
    <xs:sequence>
        <xs:element name="OpticalChecks" minOccurs="0" type="wf:type.workflow.scenario.checks"/>
        <xs:element name="ElectronicChecks" minOccurs="0" type="wf:type.workflow.scenario.checks"/>
        <xs:element name="CombinedChecks" minOccurs="0" type="wf:type.workflow.scenario.checks"/>
        <xs:element name="CrossDocumentChecks" minOccurs="0" type="wf:type.workflow.scenario.checks"/>
        <xs:element name="SealChecks" minOccurs="0" type="wf:type.workflow.scenario.checks"/>
        <xs:element name="AggregatedChecks" minOccurs="0" type="wf:type.workflow.scenario.checks"/>
    </xs:sequence>
    <xs:attribute name="preset" type="xs:string" use="required"/>
</xs:complexType>
```

6.5.5. type.workflow.scenario.checks

Workflow preset customization for the corresponding type of checks.
6.5.5.1. Attributes

None.

6.5.5.2. Elements

The type.workflow.scenario.checks elements are given in Table 6.39.

<table>
<thead>
<tr>
<th>Element name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Check*</td>
<td>wf:type.workflow.scenario.check</td>
</tr>
<tr>
<td></td>
<td>Configuration of a single check.</td>
</tr>
</tbody>
</table>

Table 6.39 type.workflow.scenario.checks - elements

6.5.5.3. XSD Definition

```xml
<xs:complexType name="type.workflow.scenario.checks">
  <xs:sequence>
    <xs:element name="Check" minOccurs="0" maxOccurs="unbounded"
                 type="wf:type.workflow.scenario.check"/>
  </xs:sequence>
</xs:complexType>
```

6.5.6. type.workflow.scenario.check

Customization of the application scenario preset with respect to a particular check.

6.5.6.1. Attributes

The type.workflow.scenario.check attributes are given in Table 6.40.

<table>
<thead>
<tr>
<th>Attribute name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>xs:ID</td>
</tr>
<tr>
<td></td>
<td>The server-defined ID of the check to configure.</td>
</tr>
<tr>
<td>action</td>
<td>wf:type.workflow.scenario.check.action</td>
</tr>
<tr>
<td></td>
<td>Determines whether the check should be executed or skipped.</td>
</tr>
<tr>
<td>condition?</td>
<td>xs:IDREF</td>
</tr>
<tr>
<td></td>
<td>Reference to a condition. Controls under which circumstances this check configuration is applicable. MUST be considered fulfilled if empty or attribute was omitted.</td>
</tr>
</tbody>
</table>

Table 6.40 type.workflow.scenario.check - attributes

6.5.6.2. Elements

None.

6.5.6.3. XSD Definition

```xml
<xs:complexType name="type.workflow.scenario.check">
  <xs:attribute name="id" type="xs:ID" use="required"/>
  <xs:attribute name="action" type="wf:type.workflow.scenario.check.action" use="required"/>
  <xs:attribute name="condition" type="xs:IDREF"/>
</xs:complexType>
```
6.5.7. type.workflow.scenario.check.action

Determines whether a check is executed or not (subject to conditional application). Derived from xs:string.

6.5.7.1. Values

The type.workflow.scenario.check.action values are given in Table 6.41.

<table>
<thead>
<tr>
<th>Element name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>exec*</td>
<td>The check MUST be performed if it is technically possible.</td>
</tr>
<tr>
<td>skin</td>
<td>The check MUST be skipped.</td>
</tr>
<tr>
<td>eval</td>
<td>The check MUST be performed if it is technically possible but the check result MUST NOT influence the overall result.</td>
</tr>
</tbody>
</table>

Table 6.41 type.workflow.scenario.check.action - values

6.5.7.2. XSD Definition

```xml
<xs:simpleType name="type.workflow.scenario.check.action">
  <xs:restriction base="xs:string">
    <xs:enumeration value="exec"/>
    <xs:enumeration value="skip"/>
    <xs:enumeration value="eval"/>
  </xs:restriction>
</xs:simpleType>
```

6.5.8. type.workflow.versions

List of version information for the requested [BSI TR-03135] and HLDC feedback XML namespaces. The default values are given in Table 6.42:

<table>
<thead>
<tr>
<th>Namespace</th>
<th>Schema version</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://trdoccheck.bsi.bund.de/dc">http://trdoccheck.bsi.bund.de/dc</a></td>
<td>1</td>
</tr>
<tr>
<td><a href="http://trdoccheck.bsi.bund.de/dce/4">http://trdoccheck.bsi.bund.de/dce/4</a></td>
<td>1</td>
</tr>
<tr>
<td><a href="http://trdoccheck.bsi.bund.de/dco/4">http://trdoccheck.bsi.bund.de/dco/4</a></td>
<td>1</td>
</tr>
<tr>
<td><a href="http://trdoccheck.bsi.bund.de/dcc/4">http://trdoccheck.bsi.bund.de/dcc/4</a></td>
<td>1</td>
</tr>
<tr>
<td><a href="http://trdoccheck.bsi.bund.de/hldc/workflow/2">http://trdoccheck.bsi.bund.de/hldc/workflow/2</a></td>
<td>Same as workflow definition.</td>
</tr>
</tbody>
</table>

Table 6.42 type.workflow.versions

6.5.8.1. Attributes

None.

6.5.8.2. Elements

The type.workflow.versions elements are given in Table 6.43.
Table 6.43 type.workflow.versions - elements

<table>
<thead>
<tr>
<th>Element name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SchemaVersion*</td>
<td>wf:type.workflow.versions.version</td>
</tr>
<tr>
<td></td>
<td>Default: see above</td>
</tr>
<tr>
<td></td>
<td>Version information.</td>
</tr>
</tbody>
</table>

6.5.8.3. XSD Definition

```xml
<xs:complexType name="type.workflow.versions">
  <xs:sequence>
    <xs:element name="SchemaVersion" minOccurs="0" maxOccurs="unbounded"
                 type="wf:type.workflow.versions.version" />
  </xs:sequence>
</xs:complexType>
```

6.5.9. type.workflow.versions.version

Requested XML schema version for a particular namespace.

6.5.9.1. Attributes

The type.workflow.versions.version attributes are given in Table 6.44.

<table>
<thead>
<tr>
<th>Attribute name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>namespace</td>
<td>xs:string</td>
</tr>
<tr>
<td></td>
<td>The namespace identifier e.g. [<a href="http://trdoccheck.bsi.bund.de/dc/3">http://trdoccheck.bsi.bund.de/dc/3</a>]</td>
</tr>
<tr>
<td>schemaVersion</td>
<td>xs:decimal</td>
</tr>
<tr>
<td></td>
<td>The schema version of the namespace, e.g. 1</td>
</tr>
</tbody>
</table>

Table 6.44 type.workflow.versions.version - attributes

6.5.9.2. Elements

None.

6.5.9.3. XSD Definition

```xml
<xs:complexType name="type.workflow.versions.version">
  <xs:attribute name="namespace" type="xs:string" use="required" />
  <xs:attribute name="schemaVersion" type="xs:decimal" use="required" />
</xs:complexType>
```

6.5.10. type.workflow.conditions

List of conditions that MAY be used in other parts of the workflow definition to control workflow execution.

6.5.10.1. Attributes

None.

6.5.10.2. Elements

The elements MAY appear in any order.

The type.workflow.conditions elements are given in Table 6.45.
<table>
<thead>
<tr>
<th>Element name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MRZCondition*</td>
<td>Conditions which evaluate data from the optical MRZ or, if CAN is used, from DG1</td>
</tr>
<tr>
<td>HasCANCondition*</td>
<td>Conditions which evaluate if a CAN was detected.</td>
</tr>
<tr>
<td>HasChipCondition*</td>
<td>Conditions which evaluate if an electronic chip was detected.</td>
</tr>
<tr>
<td>ParameterCondition*</td>
<td>Conditions whose fulfillment is controlled by the application.</td>
</tr>
<tr>
<td>AndCondition*</td>
<td>Conditions that constitute a logical conjunction of two conditions.</td>
</tr>
<tr>
<td>OrCondition*</td>
<td>Conditions that constitute a logical disjunction of two conditions.</td>
</tr>
<tr>
<td>NotCondition*</td>
<td>Conditions that negate the result of another condition.</td>
</tr>
<tr>
<td>XMLCondition*</td>
<td>Conditions based on XPath expressions.</td>
</tr>
</tbody>
</table>

Table 6.45 type.workflow.versions.version - elements

### 6.5.10.3. XSD Definition

```xml
<xs:complexType name="type.workflow.conditions">  
  <xs:choice minOccurs="0" maxOccurs="unbounded">  
    <xs:element name="MRZCondition"  
      type="wf:type.workflow.conditions.mrz" />  
    <xs:element name="HasCANCondition"  
      type="wf:type.workflow.conditions.hascan" />  
    <xs:element name="HasChipCondition"  
      type="wf:type.workflow.conditions.haschip" />  
    <xs:element name="ParameterCondition"  
      type="wf:type.workflow.conditions.parameter" />  
    <xs:element name="AndCondition"  
      type="wf:type.workflow.conditions.and" />  
    <xs:element name="OrCondition"  
      type="wf:type.workflow.conditions.or" />  
    <xs:element name="NotCondition"  
      type="wf:type.workflow.conditions.not" />  
    <xs:element name="XMLCondition"  
      type="wf:type.workflow.conditions.xml" />  
  </xs:choice>  
</xs:complexType>
```

### 6.5.11. type.workflow.conditions.mrz

Compares data from the optical MRZ to the specified data. The condition is fulfilled if the data match or if type, issuer and documentNumber are empty. If Password Authenticated Connection Establishment (PACE) with CAN is performed, the MRZ data is taken from DG1.

#### 6.5.11.1. Attributes

The type.workflow.conditions.mrz attributes are given in Table 6.46.
**Attribute name** | **Description**  
---|---  
id | xs:ID  
| User-defined unique ID of the condition. Serves as reference.  
type? | xs:string  
| Perl-compatible regular expression (PCRE) to match the document type. Fill characters MUST be ignored. MUST match if empty or attribute was omitted.  
issuer? | xs:string  
| PCRE to match the issuer. Fill characters MUST be ignored. MUST match if empty or attribute was omitted.  
issuer? | xs:string  
| PCRE to match the document number. Fill characters MUST be ignored. MUST match if empty or attribute was omitted.  

**Table 6.46** type.workflow.conditions.mrz - attributes

### 6.5.11.2. Elements
None.

### 6.5.11.3. XSD Definition

```xml
<xs:complexType name="type.workflow.conditions.mrz">
  <xs:attribute name="id" type="xs:ID" use="required" />
  <xs:attribute name="type" type="xs:string" />
  <xs:attribute name="issuer" type="xs:string" />
  <xs:attribute name="documentNumber" type="xs:string" />
</xs:complexType>
```

### 6.5.12. type.workflow.conditions.hascan
Condition which is fulfilled if a card access number (CAN) is detected on the document.

#### 6.5.12.1. Attributes
The type.workflow.conditions.hascan attributes are given in **Table 6.47**.

| Attribute name | Description  
---|---  
id | xs:ID  
| User-defined unique ID of the condition. Serves as reference.  

**Table 6.47** type.workflow.conditions.hascan - attributes

#### 6.5.12.2. Elements
None.

#### 6.5.12.3. XSD Definition

```xml
<xs:complexType name="type.workflow.conditions.hascan">
  <xs:attribute name="id" type="xs:ID" use="required" />
</xs:complexType>
```
6.5.13. type.workflow.conditions.parameter
Condition whose fulfillment is controlled by the application. The application can override the value of the condition via `beginWorkflow`.

6.5.13.1. Attributes
The `type.workflow.conditions.parameter` attributes are given in Table 6.48.

<table>
<thead>
<tr>
<th>Attribute name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>xs:ID</td>
</tr>
<tr>
<td></td>
<td>User-defined unique ID of the condition. Serves as reference.</td>
</tr>
<tr>
<td>default</td>
<td>xs:boolean</td>
</tr>
<tr>
<td></td>
<td>Default value of the condition.</td>
</tr>
</tbody>
</table>

Table 6.48 type.workflow.conditions.parameter - attributes

6.5.13.2. Elements
None.

6.5.13.3. XSD Definition

```xml
<xs:complexType name="type.workflow.conditions.parameter">
  <xs:attribute name="id" type="xs:ID" use="required"/>
  <xs:attribute name="default" type="xs:boolean" use="required"/>
</xs:complexType>
```

6.5.14. type.workflow.conditions.and
Constitutes a logical conjunction of multiple conditions. The condition is fulfilled if and only if all referenced conditions are fulfilled.

6.5.14.1. Attributes
The `type.workflow.conditions.and` attributes are given in Table 6.49.

<table>
<thead>
<tr>
<th>Attribute name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>xs:ID</td>
</tr>
<tr>
<td></td>
<td>Unique ID of the condition. Serves as reference.</td>
</tr>
</tbody>
</table>

Table 6.49 type.workflow.conditions.and - attributes

6.5.14.2. Elements
The `type.workflow.conditions.and` elements are given in Table 6.50.

<table>
<thead>
<tr>
<th>Element name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>c+</td>
<td>xs:IDREF</td>
</tr>
<tr>
<td></td>
<td>Reference to a condition. MUST be considered fulfilled if empty.</td>
</tr>
</tbody>
</table>

Table 6.50 type.workflow.conditions.and - elements

6.5.14.3. XSD Definition

```xml
<xs:complexType name="type.workflow.conditions.and">
  <xs:sequence>
    <xs:element name="c" type="xs:IDREF" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
```
6.5.15. type.workflow.conditions.or

Constitutes a logical conjunction of multiple conditions. The condition is fulfilled if and only if all referenced conditions are fulfilled.

6.5.15.1. Attributes

The type.workflow.conditions.or attributes are given in Table 6.51.

<table>
<thead>
<tr>
<th>Attribute name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>xs:ID</td>
</tr>
<tr>
<td></td>
<td>User-defined unique ID of the condition which is used as reference.</td>
</tr>
</tbody>
</table>

Table 6.51 type.workflow.conditions.or - attributes

6.5.15.2. Elements

The type.workflow.conditions.or elements are given in Table 6.52.

<table>
<thead>
<tr>
<th>Element name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>c+</td>
<td>xs:IDREF</td>
</tr>
<tr>
<td></td>
<td>Reference to a condition. MUST be considered fulfilled if empty.</td>
</tr>
</tbody>
</table>

Table 6.52 type.workflow.conditions.or - elements

6.5.15.3. XSD Definition

```xml
<xsd:complexType name="type.workflow.conditions.or">
  <xsd:sequence>
    <xsd:element name="c" type="xs:IDREF" maxOccurs="unbounded" />
  </xsd:sequence>
  <xsd:attribute name="id" type="xs:ID" use="required" />
</xsd:complexType>
```

6.5.16. type.workflow.conditions.not

Constitutes a logical negation of another condition. The condition is fulfilled if and only if the referenced condition is not fulfilled

6.5.16.1. Attributes

The type.workflow.conditions.not attributes are given in Table 6.53.

<table>
<thead>
<tr>
<th>Attribute name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>xs:ID</td>
</tr>
<tr>
<td></td>
<td>User-defined unique ID of the condition which is used as reference.</td>
</tr>
<tr>
<td>c</td>
<td>xs:IDREF</td>
</tr>
<tr>
<td></td>
<td>Reference to a condition. MUST be considered fulfilled if empty.</td>
</tr>
</tbody>
</table>

Table 6.53 type.workflow.conditions.not - attributes

6.5.16.2. Elements

None.
6.5.16.3. XSD Definition

```
<xs:complexType name="type.workflow.conditions.not">
  <xs:attribute name="id" type="xs:ID" use="required" />
  <xs:attribute name="c" type="xs:IDREF" use="required"/>
</xs:complexType>
```

6.5.17. type.workflow.conditions.xml

Compares the result of an XPath 1.0 expression with a specified value. If XPath result text and value match, the condition is fulfilled.

6.5.17.1. Attributes

The type.workflow.conditions.xml attributes are given in Table 6.54.

<table>
<thead>
<tr>
<th>Attribute name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>xs:ID</td>
</tr>
<tr>
<td></td>
<td>User-defined unique ID of the condition. Serves as reference.</td>
</tr>
<tr>
<td>from</td>
<td>xs:IDREF</td>
</tr>
<tr>
<td></td>
<td>Reference to a data element.</td>
</tr>
<tr>
<td>profile</td>
<td>xs:string</td>
</tr>
<tr>
<td></td>
<td>Logging profile on which the evaluation of this condition is based.</td>
</tr>
<tr>
<td>path</td>
<td>xs:string</td>
</tr>
<tr>
<td></td>
<td>XPath 1.0 expression which selects a node in the XML. If XPath expression matches multiple nodes, the comparison of this condition MUST consider the first matching node only. The comparison MUST only consider the inner text of the respective node.</td>
</tr>
<tr>
<td>value</td>
<td>xs:string</td>
</tr>
<tr>
<td></td>
<td>The comparative value.</td>
</tr>
</tbody>
</table>

Table 6.54 type.workflow.conditions.xml - attributes

6.5.17.2. Elements

None.

6.5.17.3. XSD Definition

```
<xs:complexType name="type.workflow.conditions.xml" />
  <xs:attribute name="id" type="xs:ID" use="required" />
  <xs:attribute name="from" type="xs:IDREF" use="required" />
  <xs:attribute name="profile" type="xs:string" use="required"/>
  <xs:attribute name="path" type="xs:string" use="required" />
  <xs:attribute name="value" type="xs:string" use="required" />
</xs:complexType>
```

6.5.18. type.workflow.aggregation

Configuration of check process with respect to the aggregation of check results.

6.5.18.1. Attributes

The type.workflow.aggregation attributes are given in Table 6.55.
### Attributes

<table>
<thead>
<tr>
<th>Attributes name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>condition?</td>
<td>xs:IDREF</td>
</tr>
<tr>
<td></td>
<td>Reference to a condition. Controls whether this configuration is applicable. MUST be considered fulfilled if empty or attribute was omitted.</td>
</tr>
</tbody>
</table>

**Table 6.55** type.workflow.aggregation - attributes

### 6.5.18.2. Elements

The type.workflow.aggregation elements are given in **Table 6.56**.

<table>
<thead>
<tr>
<th>Element name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReadSequence</td>
<td>wf:type.workflow.aggregation.readseq</td>
</tr>
<tr>
<td></td>
<td>A sequence of the reading process.</td>
</tr>
</tbody>
</table>

**Table 6.56** type.workflow.aggregation - elements

### 6.5.18.3. XSD Definition

```xml
<xs:complexType name="type.workflow.aggregation">
  <xs:sequence>
    <xs:element name="ReadSequence" type="wf:type.workflow.aggregation.readseq"/>
  </xs:sequence>
  <xs:attribute name="condition" type="xs:IDREF"/>
</xs:complexType>
```

### 6.5.19. type.workflow.aggregation.readseq

The reading sequence.

#### 6.5.19.1. Attributes

None.

#### 6.5.19.1.1. Elements

The type.workflow.aggregation.readseq elements are given in **Table 6.56**.

<table>
<thead>
<tr>
<th>Element name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AggregatedDocumentCheckResult</td>
<td>wf:type.workflow.readseq.check</td>
</tr>
<tr>
<td></td>
<td>A sequence of checks.</td>
</tr>
</tbody>
</table>

**Table 6.57** type.workflow.aggregation.readseq - elements

### 6.5.19.1.2. XSD Definition

```xml
<xs:complexType name="type.workflow.aggregation.readseq">
  <xs:choice minOccurs="0" maxOccurs="unbounded">
    <xs:element name="AggregatedDocumentCheckResult" type="wf:type.workflow.readseq.check"/>
  </xs:choice>
</xs:complexType>
```

### 6.5.19.2. type.workflow.electronic

Configuration of the electronic reading and check process with respect to access protocols and the reading sequence.
6.5.19.2.1. Attributes

The `type.workflow.electronic` attributes are given in Table 6.58.

<table>
<thead>
<tr>
<th>Attribute name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>condition?</td>
<td>xs:IDREF</td>
</tr>
<tr>
<td>Reference to a condition. Controls whether this configuration is applicable. MUST be considered fullfilled if empty or attribute was omitted.</td>
<td></td>
</tr>
</tbody>
</table>

Table 6.58 `type.workflow.electronic` - attributes

6.5.19.2.2. Elements

The `type.workflow.electronic` elements are given in Table 6.59.

<table>
<thead>
<tr>
<th>Element name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AccessConfiguration?</td>
<td><code>wf:type.workflow.electronic.access</code></td>
</tr>
<tr>
<td>Conditional configuration of access protocols (Basic Access Control (BAC), PACE, PLAIN). MAY override server-side configuration.</td>
<td></td>
</tr>
<tr>
<td>EACConfiguration?</td>
<td><code>wf:type.workflow.electronic.eac</code></td>
</tr>
<tr>
<td>Ronditional configuration of the extended access protocols (EAC) after PACE. MAY override server-side configuration.</td>
<td></td>
</tr>
<tr>
<td>ChipConfiguration?</td>
<td><code>wf:type.workflow.electronic.chip</code></td>
</tr>
<tr>
<td>Conditional timeouts for chip detection.</td>
<td></td>
</tr>
<tr>
<td>ReadSequence?</td>
<td><code>wf:type.workflow.electronic.readseq</code></td>
</tr>
<tr>
<td>Definition of the electronic reading sequence. Determines which electronic data is read in which order and when to perform the electronic checks.</td>
<td></td>
</tr>
</tbody>
</table>

Table 6.59 `type.workflow.electronic` - elements

6.5.19.2.3. XSD Definition

```xml
<xs:complexType name="type.workflow.electronic">
  <xs:sequence>
    <xs:element name="AccessConfiguration" minOccurs="0" type="wf:type.workflow.electronic.access" />
    <xs:element name="EACConfiguration" minOccurs="0" type="wf:type.workflow.electronic.eac" />
    <xs:element name="ChipConfiguration" minOccurs="0" type="wf:type.workflow.electronic.chip" />
    <xs:element name="ReadSequence" minOccurs="0" type="wf:type.workflow.electronic.readseq" />
  </xs:sequence>
  <xs:attribute name="condition" type="xs:IDREF" />
</xs:complexType>
```

6.5.19.3. `type.workflow.electronic.access`

List of conditional configurations of the access protocol order (BAC, PACE, PLAIN).

6.5.19.3.1. Attributes

None.

6.5.19.3.2. Elements

The `type.workflow.electronic.access` elements are given in Table 6.60.
### 6.5.19.3.3. XSD Definition

```xml
<xs:complexType name="type.workflow.electronic.access">
  <xs:sequence>
    <xs:element name="ProtocolOrder" minOccurs="0" maxOccurs="unbounded"
                type="wf:type.workflow.electronic.access.order"/>
  </xs:sequence>
</xs:complexType>
```

### 6.5.19.4. type.workflow.electronic.access.order

Conditional configurations of the access protocol order.

#### 6.5.19.4.1. Attributes

The `type.workflow.electronic.access.order` attributes are given in [Table 6.61](#).

<table>
<thead>
<tr>
<th>Attribute name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>condition?</td>
<td>xs:IDREF</td>
</tr>
<tr>
<td>order</td>
<td>xs:string</td>
</tr>
</tbody>
</table>

- **condition?**
  - Reference to a condition. MUST be considered fulfilled if empty or attribute was omitted. If condition uses the special value `DEFAULT`, this protocol order applies if no other condition is fulfilled. If multiple `DEFAULT`-or-`order` exist, the first one is used.

- **order**
  - Comma separated list of access protocol identifiers from the following list:
    - PLAIN
    - BAC
    - PACE
  - The access protocols MUST be executed in the specified order until the first one succeeds.

[Table 6.61](#): `type.workflow.electronic.access.order` - attributes

#### 6.5.19.4.2. Elements

None.

#### 6.5.19.4.3. XSD Definition

```xml
<xs:complexType name="type.workflow.electronic.access.order">
  <xs:attribute name="condition" type="xs:IDREF"/>
  <xs:attribute name="order" type="xs:string" use="required"/>
</xs:complexType>
```

### 6.5.19.5. type.workflow.electronic.eac

List of conditionally executed extended access protocols after PACE.

#### 6.5.19.5.1. Attributes

None.
6.5.19.5.2. Elements
The type.workflow.electronic.eac elements are given in Table 6.62.

<table>
<thead>
<tr>
<th>Element name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AuthenticationAfterPACE*</td>
<td><strong>wf</strong>:type.workflow.electronic.eac.authafterpace</td>
</tr>
<tr>
<td></td>
<td>Conditional configuration of an extended access protocol after PACE.</td>
</tr>
</tbody>
</table>

Table 6.62 type.workflow.electronic.eac - elements

6.5.19.5.3. XSD Definition

```xml
<xs:complexType name="type.workflow.electronic.eac">
  <xs:sequence>
    <xs:element name="AuthenticationAfterPACE" minOccurs="0" maxOccurs="unbounded" type="wf:type.workflow.electronic.eac.authafterpace" />
  </xs:sequence>
</xs:complexType>
```

6.5.19.6. type.workflow.electronic.eac.authafterpace
Conditional configuration of an extended access protocol (EAC).

6.5.19.6.1. Attributes
The type.workflow.electronic.eac.authafterpace attributes are given in Table 6.63.

<table>
<thead>
<tr>
<th>Attribute name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>condition?</td>
<td>xs:IDREF</td>
</tr>
<tr>
<td></td>
<td>Reference to a condition. MUST be considered fullfilled if empty or attribute was omitted.</td>
</tr>
<tr>
<td>method</td>
<td><strong>wf</strong>:type.workflow.electronic.eac.authafterpace.method</td>
</tr>
<tr>
<td></td>
<td>The extended access protocol after PACE.</td>
</tr>
</tbody>
</table>

Table 6.63 type.workflow.electronic.eac.authafterpace - attributes

6.5.19.6.2. Elements
None.

6.5.19.6.3. XSD Definition

```xml
<xs:complexType name="type.workflow.electronic.eac.authafterpace">
  <xs:attribute name="condition" type="xs:IDREF" />
  <xs:attribute name="method" type="wf:type.workflow.electronic.eac.authafterpace.method" use="required" />
</xs:complexType>
```

6.5.19.7. type.workflow.electronic.eac.authafterpace.method
Allowed extended access protocols after PACE. Derived from xs:string.

6.5.19.7.1. Values
The type.workflow.electronic.eac.authafterpace.method values are given in Table 6.64.
### Table 6.64: type.workflow.electronic.eac.authafterpace.method - values

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EAC2_AUTO</td>
<td>EAC2-compliant Chip Authentication (CA) and Terminal Authentication (TA) based on the version stored in EF.CardAccess. PACE is executed with CHAT. TA is executed with public key hash derived from PACE. Expect SAC document, if EF.CardAccess does not contain any TA/CA version information. After PACE, EAC1 will be performed. For documents with CAN, the document number from DG1 is used for TA static binding.</td>
</tr>
<tr>
<td>EAC2_CA1</td>
<td>EAC2-compliant CA v1 followed by TA v1. PACE is executed with CHAT. TA is executed with public key hash derived from PACE.</td>
</tr>
<tr>
<td>EAC2_TA2</td>
<td>EAC2-compliant TA v2 followed by CA v2. PACE is executed with CHAT. TA is executed with public key hash derived from PACE.</td>
</tr>
<tr>
<td>EAC1_CHAT</td>
<td>EAC1.11-compliant CA and TA. PACE is executed with CHAT. TA is executed with document number from MRZ.</td>
</tr>
<tr>
<td>EAC1</td>
<td>EAC1.11-compliant CA and TA. PACE is executed without CHAT (CAR is read from EV.CVCA). TA is executed with document number from MRZ.</td>
</tr>
</tbody>
</table>

#### 6.5.19.7.2. XSD Definition

```xml
<xs:simpleType name="type.workflow.electronic.eac.authafterpace.method">
  <xs:restriction base="xs:string">
    <xs:enumeration value="EAC2_AUTO"/>
    <xs:enumeration value="EAC2_CA1"/>
    <xs:enumeration value="EAC2_TA2"/>
    <xs:enumeration value="EAC1_CHAT"/>
    <xs:enumeration value="EAC1"/>
  </xs:restriction>
</xs:simpleType>
```

#### 6.5.19.8. type.workflow.electronic.chip

List of conditional timeout configurations for chip detection.

#### 6.5.19.8.1. Attributes

None.

#### 6.5.19.8.2. Elements

The `type.workflow.electronic.chip` elements are given in Table 6.65.

<table>
<thead>
<tr>
<th>Element name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WaitForChip*</td>
<td>wf: type.workflow.electronic.chip.waitforchip Configuration of a conditional chip detection timeout.</td>
</tr>
</tbody>
</table>

#### 6.5.19.8.3. XSD Definition

```xml
<xs:complexType name="type.workflow.electronic.chip">
  <xs:attribute name="ForChip" type="xs:string" use="optional"/>
  <xs:attribute name="ForTimeout" type="xs:string" use="optional"/>
</xs:complexType>
```
6.5.19.9. **type.workflow.electronic.chip.waitforchip**

Conditional chip detection timeout. Before starting the electronic reading process, the server MUST wait for the specified time period unless the chip is detected successfully earlier. If omitted, the server MAY start reading the chip immediately.

6.5.19.9.1. **Attributes**

The `type.workflow.electronic.chip.waitforchip` elements are given in Table 6.66.

<table>
<thead>
<tr>
<th>Attribute name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>condition?</td>
<td>xs:IDREF</td>
</tr>
<tr>
<td></td>
<td>Reference to a condition. MUST be considered fulfilled if empty or attribute was omitted.</td>
</tr>
<tr>
<td>timeInMs</td>
<td>xs:int</td>
</tr>
<tr>
<td></td>
<td>Chip detection timeout in milliseconds.</td>
</tr>
<tr>
<td>intervalInMs?</td>
<td>xs:int</td>
</tr>
<tr>
<td></td>
<td>Default: 500</td>
</tr>
<tr>
<td></td>
<td>Polling interval in milliseconds. During <code>timeInMs</code>, the server SHOULD wait at least <code>intervalInMs</code> milliseconds between attempts to detect the chip.</td>
</tr>
</tbody>
</table>

Table 6.66: `type.workflow.electronic.chip.waitforchip` - attributes

6.5.19.9.2. **Elements**

None.

6.5.19.9.3. **XSD Definition**

```xml
<xs:complexType name="type.workflow.electronic.chip.waitforchip">
    <xs:attribute name="condition" type="xs:IDREF" />
    <xs:attribute name="timeInMs" type="xs:int" use="required" />
    <xs:attribute name="intervalInMs" type="xs:int" default="500" />
</xs:complexType>
```

6.5.19.10. **type.workflow.electronic.readseq**

List of read and check operations to perform. The execution order corresponds to the order of the definition.

6.5.19.10.1. **Attributes**

None.

6.5.19.10.2. **Elements**

The elements MAY appear in any order.

The `type.workflow.electronic.readseq` elements are given in Table 6.67.

<table>
<thead>
<tr>
<th>Element name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Datagroup*</td>
<td><code>wf:type.workflow.electronic.readseq.datagroup</code></td>
</tr>
<tr>
<td></td>
<td>Reads an electronic data group.</td>
</tr>
</tbody>
</table>
Element name | Description
--- | ---
ElectronicCheck* | \( \text{wf:} \text{type.workflow.readseq.check} \)
                | Performs an electronic document check according to [BSI TR-03135].
DefectInfo*     | \( \text{wf:} \text{type.workflow.electronic.readseq.defectinfo} \)
                | Reads known defects for the document.
ChipDetection*  | \( \text{wf:} \text{type.workflow.electronic.readseq.chipdetection} \)
                | Checks for the presence of a radio-frequency identification (RFID) chip
                | in the reading field. The result can be acquired by a text feedback element.

Table 6.67 \text{type.workflow.electronic.readseq} - elements.

### 6.5.19.10.3. XSD Definition

```xml
<xs:complexType name="type.workflow.electronic.readseq">
    <xs:choice minOccurs="0" maxOccurs="unbounded">
        <xs:element name="Datagroup" type="wf:type.workflow.electronic.readseq.datagroup" />
        <xs:element name="ElectronicCheck" type="wf:type.workflow.readseq.check" />
        <xs:element name="DefectInfo" type="wf:type.workflow.electronic.readseq.defectinfo" />
        <xs:element name="ChipDetection" type="wf:type.workflow.electronic.readseq.chipdetection" />
    </xs:choice>
</xs:complexType>
```

### 6.5.19.11. \text{type.workflow.electronic.readseq.datagroup}

Configuration of conditional reading of data groups.

#### 6.5.19.11.1. Attributes

The \text{type.workflow.electronic.readseq.datagroup} attributes are given in Table 6.68.

<table>
<thead>
<tr>
<th>Attribute name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>( \text{xs:} \text{ID} )</td>
</tr>
<tr>
<td></td>
<td>User-defined unique ID. Serves as reference.</td>
</tr>
<tr>
<td>application</td>
<td>( \text{wf:} \text{type.workflow.electronic.readseq.datagroup.application} )</td>
</tr>
<tr>
<td></td>
<td>The document application from which the data group is to be read.</td>
</tr>
<tr>
<td>number</td>
<td>( \text{xs:int} )</td>
</tr>
<tr>
<td></td>
<td>The number of the data group to read.</td>
</tr>
<tr>
<td>condition?</td>
<td>( \text{xs:IDREF} )</td>
</tr>
<tr>
<td></td>
<td>Reference to a condition. Controls whether this configuration is applicable.</td>
</tr>
<tr>
<td></td>
<td>MUST be considered fulfilled if empty or omitted.</td>
</tr>
<tr>
<td>required?</td>
<td>( \text{xs:boolean} )</td>
</tr>
<tr>
<td></td>
<td>Default: false</td>
</tr>
<tr>
<td></td>
<td>Specifies whether reading this data group is imperative for the respective</td>
</tr>
<tr>
<td></td>
<td>application. If not, failure to read the data group MUST NOT negatively</td>
</tr>
<tr>
<td></td>
<td>affect corresponding data integrity checks.</td>
</tr>
</tbody>
</table>

Table 6.68 \text{type.workflow.electronic.readseq.datagroup} - attributes
6.5.19.12.1. Values

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>International Civil Aviation Organization (ICAO)</td>
<td>ICAO ePassport datagroups</td>
</tr>
<tr>
<td>GermanID</td>
<td>German Electronic Identity Document (eID) datagroups</td>
</tr>
</tbody>
</table>

Table 6.69 type.workflow.electronic.readseq.datagroup.application - values

6.5.19.13. type.workflow.electronic.readseq.defectinfo

Conditional reading of the known defects regarding the current document.

6.5.19.13.1. Attributes

<table>
<thead>
<tr>
<th>Attribute name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>User-defined unique ID. Serves as reference.</td>
</tr>
<tr>
<td>condition?</td>
<td>Reference to a condition. Controls whether this configuration is applicable. MUST be considered fullfilled if empty or attribute was omitted.</td>
</tr>
</tbody>
</table>

Table 6.70 type.workflow.electronic.readseq.defectinfo - attributes

6.5.19.13.2. Elements

None.
6.5.19.13.3. XSD Definition

```xml
<xs:complexType name="type.workflow.electronic.readseq.defectinfo">
  <xs:attribute name="id" type="xs:ID" use="required"/>
  <xs:attribute name="condition" type="xs:IDREF"/>
</xs:complexType>
```

6.5.19.14. type.workflow.electronic.readseq.chipdetection

Checking for the presence of an RFID chip in the reading field.

6.5.19.14.1. Attributes

The type.workflow.electronic.readseq.chipdetection attributes are given in Table 6.71.

<table>
<thead>
<tr>
<th>Attribute name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>xs:ID</td>
</tr>
<tr>
<td>condition?</td>
<td>xs:IDREF</td>
</tr>
</tbody>
</table>

Table 6.71 type.workflow.electronic.readseq.chipdetection - attributes

6.5.19.14.2. Elements

None.

6.5.19.14.3. XSD Definition

```xml
<xs:complexType name="type.workflow.electronic.readseq.chipdetection">
  <xs:attribute name="id" type="xs:ID" use="required"/>
  <xs:attribute name="condition" type="xs:IDREF"/>
</xs:complexType>
```

6.5.19.15. type.workflow.electronic.readseq.elementaryfile

Configuration of conditional reading of elementary files.

6.5.19.15.1. Attributes

The type.workflow.electronic.readseq.elementaryfile attributes are given in Table 6.72.

<table>
<thead>
<tr>
<th>Attribute name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>xs:ID</td>
</tr>
<tr>
<td>name</td>
<td>wf:type.workflow.electronic.readseq.elementaryfile.name</td>
</tr>
<tr>
<td>condition?</td>
<td>xs:IDREF</td>
</tr>
<tr>
<td>required?</td>
<td>xs:boolean</td>
</tr>
</tbody>
</table>

Table 6.72 type.workflow.electronic.readseq.elementaryfile - attributes
6.5.19.15.2. Elements
None.

6.5.19.15.3. XSD Definition

```xml
<xs:complexType name="type.workflow.electronic.readseq.elementaryfile">
  <xs:attribute name="id" type="xs:ID" use="required" />
  <xs:attribute name="name" type="wf:type.workflow.electronic.readseq.elementaryfile.name" use="required" />
  <xs:attribute name="condition" type="xs:IDREF" />
  <xs:attribute name="required" type="xs:boolean" default="false" />
</xs:complexType>
```

6.5.19.16. `type.workflow.electronic.readseq.elementaryfile.name`

Selects the elementary file for reading by its name.

6.5.19.16.1. Values

The `type.workflow.electronic.readseq.elementaryfile.name` values are given in Table 6.73.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ef_sod</td>
<td>EF.SoD</td>
</tr>
<tr>
<td>ef_card_security</td>
<td>EF.CardSecurity</td>
</tr>
<tr>
<td>ef_chip_security</td>
<td>EF.ChipSecurity</td>
</tr>
<tr>
<td>ef_com</td>
<td>EF.COM</td>
</tr>
<tr>
<td>ef_cvca</td>
<td>EF.CVCA</td>
</tr>
<tr>
<td>ef_card_access</td>
<td>EF.CardAccess</td>
</tr>
<tr>
<td>ef_atr_info</td>
<td>EF.ATR_INFO</td>
</tr>
</tbody>
</table>

Table 6.73 `type.workflow.electronic.readseq.elementaryfile.name` - values

6.5.19.16.2. XSD Definition

```xml
<xs:simpleType name="type.workflow.electronic.readseq.elementaryfile.name">
  <xs:restriction base="xs:string">
    <xs:enumeration value="ef_sod" />
    <xs:enumeration value="ef_card_security" />
    <xs:enumeration value="ef_chip_security" />
    <xs:enumeration value="ef_com" />
    <xs:enumeration value="ef_cvca" />
    <xs:enumeration value="ef_card_access" />
    <xs:enumeration value="ef_atr_info" />
  </xs:restriction>
</xs:simpleType>
```

6.5.19.17. `type.workflow.optical`

Configuration of the optical reading and check process with respect to document validity definition and the optical reading sequence.

6.5.19.17.1. Attributes

The `type.workflow.optical` attributes are given in Table 6.74.
### Attribute name | Description
--- | ---
condition? | xs:IDREF

Reference to a condition. Controls whether this configuration is applicable. MUST be considered fulfilled if empty or attribute was omitted.

**Table 6.74** type.workflow.optical - attributes

#### 6.5.19.17.2. Elements

The type.workflow.optical elements are given in »Table 6.75.

| Element name | Description |
--- | ---
DocValidityConfiguration? | wf:type.workflow.optical.validity
ReadSequence? | wf:type.workflow.optical.readseq

Definition of the optical reading sequence. Determines which optical data is read in which order and when to perform the optical checks.

**Table 6.75** type.workflow.optical - elements

#### 6.5.19.17.3. XSD Definition

```xml
<xs:complexType name="type.workflow.optical">
  <xs:sequence>
    <xs:element name="DocValidityConfiguration" minOccurs="0" type="wf:type.workflow.optical.validity" />
    <xs:element name="ReadSequence" minOccurs="0" type="wf:type.workflow.optical.readseq" />
  </xs:sequence>
  <xs:attribute name="condition" type="xs:IDREF" />
</xs:complexType>
```

#### 6.5.19.18. type.workflow.optical.validity

List of document validity configurations.

#### 6.5.19.18.1. Attributes

None.

#### 6.5.19.18.2. Elements

The type.workflow.optical.validity elements are given in »Table 6.76.

| Element name | Description |
--- | ---
DocValidity* | wf:type.workflow.optical.validity.element

Conditional document validity configuration.

**Table 6.76** type.workflow.optical.validity - elements

#### 6.5.19.18.3. XSD Definition

```xml
<xs:complexType name="type.workflow.optical.validity">
  <xs:sequence>
    <xs:element name="DocValidity" type="wf:type.workflow.optical.validity.element" minOccurs="0" maxOccurs="unbounded" />
  </xs:sequence>
</xs:complexType>
```
6.5.19.19. **type.workflow.optical.validity.element**
Conditional configuration of the document validity.

6.5.19.19.1. Attributes
The type.workflow.optical.validity.element attributes are given in Table 6.77.

<table>
<thead>
<tr>
<th>Attribute name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>condition</td>
<td>xs:IDREF Reference to a condition. Controls whether this configuration is applicable. MUST be considered fulfilled if empty. If condition uses the special value DEFAULT, this configuration applies if no other condition is fulfilled. If multiple DEFAULT-configurations exist, the first one is used.</td>
</tr>
<tr>
<td>validityInYears</td>
<td>xs:unsignedInt Document validity in years.</td>
</tr>
<tr>
<td>toleranceInDays?</td>
<td>xs:int Default: 0 Tolerance in days after document expiration for which the document is still considered valid. This value is used in the optical document check. This value can also be negative to indicate that a document is already considered as invalid before its regular expiration.</td>
</tr>
</tbody>
</table>

Table 6.77 type.workflow.optical.validity.element - attributes

6.5.19.19.2. Elements
None.

6.5.19.19.3. XSD Definition

```xml
<xs:complexType name="type.workflow.optical.validity.element">
  <xs:attribute name="condition" type="xs:IDREF" use="required" />
  <xs:attribute name="validityInYears" type="xs:unsignedInt" use="required" />
  <xs:attribute name="toleranceInDays?" type="xs:unsignedInt" default="0" />
</xs:complexType>
```

6.5.19.20. **type.workflow.optical.readseq**
List of optical read and check operations to perform. The execution order corresponds to the order of the definition.

6.5.19.20.1. Attributes
None.

6.5.19.20.2. Elements
The elements MAY appear in any order.

The type.workflow.optical.readseq elements are given in Table 6.78.

<table>
<thead>
<tr>
<th>Element name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Text*</td>
<td>wf:type.workflow.optical.readseq.element Reads a text field.</td>
</tr>
<tr>
<td>Element name</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Binary*</td>
<td>wf:type.workflow.optical.readseq.element</td>
</tr>
<tr>
<td></td>
<td>Reads a binary field.</td>
</tr>
<tr>
<td>Image*</td>
<td>wf:type.workflow.optical.readseq.element</td>
</tr>
<tr>
<td></td>
<td>Reads an image field.</td>
</tr>
<tr>
<td>OpticalCheck*</td>
<td>wf:type.workflow.readseq.check</td>
</tr>
<tr>
<td></td>
<td>Performs optical document checks according to [BSI TR-03135].</td>
</tr>
</tbody>
</table>

Table 6.78 type.workflow.optical.readseq - elements

6.5.19.20.3. XSD Definition

```xml
<xs:complexType name="type.workflow.optical.readseq">
  <xs:choice minOccurs="0" maxOccurs="unbounded">
    <xs:element name="Text" type="wf:type.workflow.optical.readseq.element"/>
    <xs:element name="Binary" type="wf:type.workflow.optical.readseq.element"/>
    <xs:element name="Image" type="wf:type.workflow.optical.readseq.element"/>
    <xs:element name="OpticalCheck" type="wf:type.workflow.readseq.check"/>
  </xs:choice>
</xs:complexType>
```

6.5.19.21. type.workflow.optical.readseq.element

Conditionally specifies which optical data is to be read.

6.5.19.21.1. Attributes

The type.workflow.optical.readseq.element attributes are given in Table 6.79.

<table>
<thead>
<tr>
<th>Attributes name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>xs:ID</td>
</tr>
<tr>
<td>field</td>
<td>xs:string</td>
</tr>
<tr>
<td>condition?</td>
<td>xs:IDREF</td>
</tr>
</tbody>
</table>

Reference to a condition. Controls whether this configuration is applicable. MUST be considered fullfilled if empty or attribute was omitted.

Table 6.79 type.workflow.optical.readseq.element - attributes

6.5.19.21.2. Elements

None.

6.5.19.21.3. XSD Definition

```xml
<xs:complexType name="type.workflow.optical.readseq.element">
  <xs:attribute name="id" type="xs:ID" use="required"/>
  <xs:attribute name="field" type="xs:string" use="required"/>
  <xs:attribute name="condition" type="xs:IDREF"/>
</xs:complexType>
```

6.5.19.22. type.workflow.combined

Configuration of the combined reading and check process.
6.5.19.22.1. Attributes

The type.workflow.combined attributes are given in Table 6.80.

<table>
<thead>
<tr>
<th>Attributes name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>condition?</td>
<td>xs:IDREF</td>
</tr>
<tr>
<td></td>
<td>Reference to a condition. Controls whether this configuration is applicable. MUST be considered fullfilled if empty or attribute was omitted.</td>
</tr>
</tbody>
</table>

Table 6.80 type.workflow.combined - attributes

6.5.19.22.2. Elements

The type.workflow.combined elements are given in Table 6.81.

<table>
<thead>
<tr>
<th>Element name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReadSequence?</td>
<td>wf:type.workflow.combined.readseq</td>
</tr>
<tr>
<td></td>
<td>Definition of the combined reading sequence. Determines which combined data is read in which order and when to perform the combined checks.</td>
</tr>
</tbody>
</table>

Table 6.81 type.workflow.combined - elements

6.5.19.22.3. XSD Definition

```xml
<xs:complexType name="type.workflow.combined">
    <xs:sequence>
        <xs:element name="ReadSequence" minOccurs="0" maxOccurs="unbounded">
            <type wf:type.workflow.combined.readseq/>
        </xs:element>
        <xs:attribute name="condition" type="xs:IDREF"/>
    </xs:sequence>
</xs:complexType>
```

6.5.19.23. type.workflow.combined.readseq

List of combined check operations to perform. The execution order corresponds to the order of the definition.

6.5.19.23.1. Attributes

None.

6.5.19.23.2. Elements

The elements MAY appear in any order.

The type.workflow.combined.readseq elements are given in Table 6.82.

<table>
<thead>
<tr>
<th>Element name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CombinedCheck*</td>
<td>wf:type.workflow.readseq.check</td>
</tr>
<tr>
<td></td>
<td>A single-document combined check.</td>
</tr>
<tr>
<td>CrossCheck*</td>
<td>wf:type.workflow.readseq.check</td>
</tr>
<tr>
<td></td>
<td>A cross-document combined check.</td>
</tr>
</tbody>
</table>

Table 6.82 type.workflow.combined.readseq - elements

6.5.19.23.3. XSD Definition

```xml
<xs:complexType name="type.workflow.combined.readseq">
    <xs:choice minOccurs="0" maxOccurs="unbounded">
        <xs:element name="CombinedCheck" type="wf:type.workflow.readseq.ccheck"/>
    </xs:choice>
</xs:complexType>
```
### 6.5.19.24. type.workflow.readseq.check

Conditional execution of a particular document check according to [BSI TR-03135].

#### 6.5.19.24.1. Attributes

The type.workflow.readseq.check attributes are given in Table 6.83.

<table>
<thead>
<tr>
<th>Attribute name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>xs:ID</td>
</tr>
<tr>
<td></td>
<td>User-defined unique ID. Serves as reference.</td>
</tr>
<tr>
<td>condition</td>
<td>xs:IDREF</td>
</tr>
<tr>
<td></td>
<td>Reference to a condition. Controls whether this configuration is applicable. MUST be considered fulfilled if empty or attribute was omitted.</td>
</tr>
</tbody>
</table>

Table 6.83 type.workflow.readseq.check - attributes

#### 6.5.19.24.2. Elements

None.

#### 6.5.19.24.3. XSD Definition

```xml
<xs:complexType name="type.workflow.readseq.check">
  <xs:attribute name="id" type="xs:ID" use="required" />
  <xs:attribute name="condition" type="xs:IDREF" />
</xs:complexType>
```

### 6.5.19.25. type.workflow.readseq.defect

Conditional execution of a particular document defect according to [BSI TR-03135].

#### 6.5.19.25.1. Attributes

The type.workflow.readseq.defect attributes are given in Table 6.84.

<table>
<thead>
<tr>
<th>Attribute name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>xs:ID</td>
</tr>
<tr>
<td></td>
<td>User-defined unique ID. Serves as reference.</td>
</tr>
<tr>
<td>condition</td>
<td>xs:IDREF</td>
</tr>
<tr>
<td></td>
<td>Reference to a condition. Controls whether this configuration is applicable. MUST be considered fulfilled if empty or attribute was omitted.</td>
</tr>
</tbody>
</table>

Table 6.84 type.workflow.readseq.defect - attributes

#### 6.5.19.25.2. Elements

None.

#### 6.5.19.25.3. XSD Definition

```xml
<xs:complexType name="type.workflow.readseq.defect">
  <xs:attribute name="id" type="xs:ID" use="required" />
  <xs:attribute name="condition" type="xs:IDREF" />
</xs:complexType>
```
6.5.19.26. type.workflow.feedback
List of feedback elements that provide data from the elements of the electronic, optical and combined reading sequences. The feedback elements are the only data that are delivered to the application. Feedback is provided in the feedback loop.

6.5.19.26.1. Attributes
None.

6.5.19.26.2. Elements
The elements MAY appear in any order.
The type.workflow.feedback elements are given in Table 6.85.

<table>
<thead>
<tr>
<th>Element name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Binary*</td>
<td>Returns binary data to the application.</td>
</tr>
<tr>
<td>Text*</td>
<td>Returns text data to the application.</td>
</tr>
<tr>
<td>Image*</td>
<td>Returns image data to the application.</td>
</tr>
<tr>
<td>XML*</td>
<td>Returns XML data to the application.</td>
</tr>
</tbody>
</table>

Table 6.85: type.workflow.feedback - elements

6.5.19.26.3. XSD Definition

```xml
<xs:complexType name="type.workflow.feedback">
  <xs:choice minOccurs="0" maxOccurs="unbounded">
    <xs:element name="Binary" type="wf:type.workflow.feedback.binary" />
    <xs:element name="Text" type="wf:type.workflow.feedback.text" />
    <xs:element name="Image" type="wf:type.workflow.feedback.image" />
    <xs:element name="XML" type="wf:type.workflow.feedback.xml" />
  </xs:choice>
</xs:complexType>
```

6.5.19.27. type.workflow.seal
Configuration of the seal reading and check process with respect to document validity definition and the seal reading sequence.

6.5.19.27.1. Attributes
The type.workflow.seal are given in Table 6.86.

<table>
<thead>
<tr>
<th>Attribute name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>condition?</td>
<td>xs:IDREF</td>
</tr>
</tbody>
</table>

Reference to a condition. Controls whether this configuration is applicable. MUST be considered fulfilled if empty or attribute was omitted.

Table 6.86: type.workflow.seal - attributes
6.5.19.27.2. Elements

The type.workflow.seal.readseq elements are given in Table 6.87.

<table>
<thead>
<tr>
<th>Element name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReadSequence?</td>
<td>wf:type.workflow.seal.readseq</td>
</tr>
<tr>
<td></td>
<td>Definition of the seal reading sequence. Determines which seal data is read in which order and when to perform the seal checks.</td>
</tr>
</tbody>
</table>

Table 6.87 type.workflow.seal - elements

6.5.19.27.3. XSD Definition

```xml
<xs:complexType name="type.workflow.seal"
  xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:sequence>
    <xs:element name="ReadSequence" minOccurs="0" type="wf:type.workflow.seal.readseq"/>
    <xs:attribute name="condition" type="xs:IDREF"/>
  </xs:sequence>
</xs:complexType>
```

6.5.19.28. type.workflow.electronic.readseq

List of read and check operations to perform. The execution order corresponds to the order of the definition.

6.5.19.28.1. Attributes

None.

6.5.19.28.2. Elements

The elements MAY appear in any order.

The type.workflow.seal.readseq elements are given in Table 6.88.

<table>
<thead>
<tr>
<th>Element name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data*</td>
<td>wf:type.workflow.seal.readseq.data</td>
</tr>
<tr>
<td></td>
<td>Reads the seal data.</td>
</tr>
<tr>
<td>Check*</td>
<td>wf:type.workflow.readseq.check</td>
</tr>
<tr>
<td></td>
<td>Performs a digital seal document check according to [BSI TR-03135].</td>
</tr>
<tr>
<td>Defect</td>
<td>wf:type.workflow.readseq.defect</td>
</tr>
<tr>
<td></td>
<td>Reads known defects for the document.</td>
</tr>
</tbody>
</table>

Table 6.88 type.workflow.electronic.readseq - elements.

6.5.19.28.3. XSD Definition

```xml
<xs:complexType name="type.workflow.seal.readseq">
  <xs:choice minOccurs="0" maxOccurs="unbounded">
    <xs:element name="Data" type="wf:type.workflow.seal.readseq.data"/>
    <xs:element name="Defect" type="wf:type.workflow.readseq.defect"/>
    <xs:element name="Check" type="wf:type.workflow.readseq.check"/>
  </xs:choice>
</xs:complexType>
```

6.5.19.29. type.workflow.seal.readseq.data

Configuration of conditional reading of data.
6.5.19.29.1. Attributes

The type.workflow.seal.readseq.data attributes are given in Table 6.89.

<table>
<thead>
<tr>
<th>Attribute name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>xs:ID</td>
</tr>
<tr>
<td>profile</td>
<td>wf:type.workflow.seal.readseq.data.profile</td>
</tr>
<tr>
<td>condition?</td>
<td>xs:IDREF</td>
</tr>
<tr>
<td>required?</td>
<td>xs:boolean</td>
</tr>
</tbody>
</table>

Table 6.89 type.workflow.seal.readseq.data - attributes

6.5.19.29.2. Elements

None.

6.5.19.29.3. XSD Definition

```xml
<xs:complexType name="type.workflow.seal.readseq.data">
  <xs:attribute name="id" type="xs:ID" use="required"/>
  <xs:attribute name="profile" type="wf:type.workflow.seal.readseq.data.profile" use="required"/>
  <xs:attribute name="condition" type="xs:IDREF"/>
  <xs:attribute name="required" type="xs:boolean"/>
</xs:complexType>
```

6.5.19.30. type.workflow.feedback.binary

Requests binary data feedback. Binary data is always returned in the binaryFeedback element. It is only available for electronic data groups.

6.5.19.30.1. Attributes

The type.workflow.feedback.binary attributes are given in Table 6.90.

<table>
<thead>
<tr>
<th>Attribute name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>xs:ID</td>
</tr>
<tr>
<td>from</td>
<td>xs:IDREF</td>
</tr>
<tr>
<td>condition?</td>
<td>xs:IDREF</td>
</tr>
</tbody>
</table>

Reference to data element. Referenced element MUST be of type type.workflow.electronic.readseq.datagroup.
Attribute name | Description
---|---
onRequestOnly? | xs:boolean

Default: false

Controls whether the feedback element is available as part of the general feedback loop via `getWorkflowFeedback` or only upon explicit request via `getWorkflowFeedbackById`.

<table>
<thead>
<tr>
<th>Attribute name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>xs:ID</td>
</tr>
</tbody>
</table>

User-defined unique ID. Serves as reference.

| from | xs:IDREF |

Reference to a data element. MUST reference any of the following:

- ICAO data group 1
- eID data group 19
- eID data group 20
- any optical text field

| condition? | xs:IDREF |

References a condition. Controls whether the feedback element is delivered. MUST be considered fullfilled if empty or attribute was omitted.

| onRequestOnly? | xs:boolean |

Default: false

Controls whether the feedback element is available as part of the general feedback loop via `getWorkflowFeedback` or only upon explicit request via `getWorkflowFeedbackById`.

### 6.5.19.30.3. XSD Definition

```xml
<xs:complexType name="type.workflow.feedback.binary">
  <xs:attribute name="id" type="xs:ID" use="required" />
  <xs:attribute name="from" type="xs:IDREF" use="required" />
  <xs:attribute name="condition" type="xs:IDREF" />
  <xs:attribute name="onRequestOnly" type="xs:boolean" default="false" />
</xs:complexType>
```

### 6.5.19.31. type.workflow.feedback.text

Requests text data feedback. Text data is always returned in the `stringFeedback` element.

#### 6.5.19.31.1. Attributes

The type.workflow.feedback.text attributes are given in Table 6.91.

<table>
<thead>
<tr>
<th>Attribute name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>xs:ID</td>
</tr>
</tbody>
</table>

User-defined unique ID. Serves as reference.

| from | xs:IDREF |

Reference to a data element. MUST reference any of the following:

- ICAO data group 1
- eID data group 19
- eID data group 20
- any optical text field

| condition? | xs:IDREF |

References a condition. Controls whether the feedback element is delivered. MUST be considered fullfilled if empty or attribute was omitted.

| onRequestOnly? | xs:boolean |

Default: false

Controls whether the feedback element is available as part of the general feedback loop via `getWorkflowFeedback` or only upon explicit request via `getWorkflowFeedbackById`.

### 6.5.19.31.2. Elements

None.
6.5.19.31.3. XSD Definition

```xml
<xs:complexType name="type.workflow.feedback.text">
  <xs:attribute name="id" type="xs:ID" use="required" />
  <xs:attribute name="from" type="xs:IDREF" use="required" />
  <xs:attribute name="condition" type="xs:IDREF" />
  <xs:attribute name="onRequestOnly" type="wf:type.yesno" default="false" />
</xs:complexType>
```

6.5.19.32. type.workflow.feedback.image

Requests image data feedback. Image data is always returned in the binaryFeedback field.

6.5.19.32.1. Attributes

The type.workflow.feedback.image attributes are given in Table 6.92.

<table>
<thead>
<tr>
<th>Attribute name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>xs:ID</td>
</tr>
<tr>
<td>from</td>
<td>xs:IDREF</td>
</tr>
<tr>
<td>format?</td>
<td>wf:type.workflow.feedback.image.format</td>
</tr>
<tr>
<td>size?</td>
<td>xs:string</td>
</tr>
<tr>
<td>preserveAspectRatio?</td>
<td>xs:boolean</td>
</tr>
<tr>
<td>index?</td>
<td>xs:unsignedInt</td>
</tr>
<tr>
<td>condition?</td>
<td>xs:IDREF</td>
</tr>
<tr>
<td>onRequestOnly?</td>
<td>xs:boolean</td>
</tr>
</tbody>
</table>

Table 6.92 type.workflow.feedback.image - attributes
6.5.19.32.2. Elements
None.

6.5.19.32.3. XSD Definition

```xml
<xs:complexType name="type.workflow.feedback.image">
  <xs:attribute name="id" type="xs:ID" use="required"/>
  <xs:attribute name="from" type="xs:IDREF" use="required"/>
  <xs:attribute name="format" type="wf:type.workflow.feedback.image.format"/>
  <xs:attribute name="size" type="xs:string"/>
  <xs:attribute name="preserveAspectRatio" type="xs:boolean" default="true"/>
  <xs:attribute name="index" type="xs:unsignedInt"/>
  <xs:attribute name="condition" type="xs:IDREF"/>
  <xs:attribute name="onRequestOnly" type="xs:boolean" default="false"/>
</xs:complexType>
```

6.5.19.33. type.workflow.feedback.xml

Requests XML-formatted feedback. XML data is always returned in the stringFeedback field.

6.5.19.33.1. Attributes

The type.workflow.feedback.xml attributes are given in Table 6.93.

<table>
<thead>
<tr>
<th>Attribute name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>xs:ID</td>
</tr>
<tr>
<td></td>
<td>Unique ID. Serves as reference.</td>
</tr>
<tr>
<td>from</td>
<td>xs:IDREF</td>
</tr>
<tr>
<td></td>
<td>Reference to a data element. MUST reference any of the following:</td>
</tr>
<tr>
<td></td>
<td>• ICAO data group 1</td>
</tr>
<tr>
<td></td>
<td>• ICAO data group 2</td>
</tr>
<tr>
<td></td>
<td>• ICAO data group 3</td>
</tr>
<tr>
<td></td>
<td>• ICAO data group 4</td>
</tr>
<tr>
<td></td>
<td>• ICAO data group 11</td>
</tr>
<tr>
<td></td>
<td>• ICAO data group 12</td>
</tr>
<tr>
<td></td>
<td>• any field of type</td>
</tr>
<tr>
<td></td>
<td>• type.workflow.electronic.readseq.defectinfo</td>
</tr>
<tr>
<td></td>
<td>• type.workflow.readseq.check</td>
</tr>
<tr>
<td>format?</td>
<td>wf:type.workflow.feedback.image.format</td>
</tr>
<tr>
<td></td>
<td>Requested image format. If omitted, the image is delivered in the original</td>
</tr>
<tr>
<td></td>
<td>format.</td>
</tr>
<tr>
<td>size?</td>
<td>xs:string</td>
</tr>
<tr>
<td></td>
<td>Requested image size in format &lt;Width&gt;x&lt;Height&gt; (e.g. 1000x800). If omitted,</td>
</tr>
<tr>
<td></td>
<td>the image is delivered in the original image size.</td>
</tr>
<tr>
<td>preserveAspectRatio?</td>
<td>xs:boolean</td>
</tr>
<tr>
<td></td>
<td>Default: true</td>
</tr>
<tr>
<td></td>
<td>Controls whether the aspect ratio of the image is preserved on scaling.</td>
</tr>
<tr>
<td>profile</td>
<td>xs:string</td>
</tr>
<tr>
<td></td>
<td>Logging profile on which the XPath evaluation is based.</td>
</tr>
</tbody>
</table>
### Attribute name | Description
--- | ---
path? | xs:string
XPath 1.0 expression to limit the returned XML data to matching nodes. If empty or omitted, the complete XML data is delivered.
condition? | xs:IDREF
References a condition. Controls whether the feedback element is delivered. MUST be considered fulfilled if empty or attribute was omitted.
onRequestOnly? | xs:boolean
Default: false
Controls whether the feedback element is available as part of the general feedback loop via `getWorkflowFeedback` or only upon explicit request via `getWorkflowFeedbackById`.
innerTextOnly? | xs:boolean
Default: false
Controls whether only the inner text of a selected node is delivered as `stringFeedback`. If yes, path MUST NOT be empty.

Table 6.93 type.workflow.feedback.xml - attributes

#### 6.5.19.33.2. Elements

None.

#### 6.5.19.33.3. XSD Definition

```xml
<xs:complexType name="type.workflow.feedback.xml">
  <xs:attribute name="id" type="xs:ID" use="required"/>
  <xs:attribute name="from" type="xs:IDREF" use="required"/>
  <xs:attribute name="format" type="wf:type.workflow.feedback.image.format"/>
  <xs:attribute name="size" type="xs:string"/>
  <xs:attribute name="preserveAspectRatio" type="xs:string"/>
  <xs:attribute name="profile" type="xs:string"/>
  <xs:attribute name="path" type="xs:string"/>
  <xs:attribute name="condition" type="xs:IDREF"/>
  <xs:attribute name="onRequestOnly" type="xs:boolean" default="false"/>
  <xs:attribute name="innerTextOnly" type="xs:boolean" default="false"/>
</xs:complexType>
```

#### 6.5.19.34. type.workflow.feedback.image.format

Denotes the binary data format of image feedback. Derived from `xs:string`.

#### 6.5.19.34.1. Values

The `type.workflow.feedback.image.format` values are given in Table 6.94.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bmp</td>
<td>Bitmap</td>
</tr>
<tr>
<td>jpeg</td>
<td>JPEG</td>
</tr>
<tr>
<td>jpeg2000</td>
<td>JPEG 2000</td>
</tr>
<tr>
<td>png</td>
<td>Portable Network Graphics (Portable Network Graphics (PNG))</td>
</tr>
<tr>
<td>wsq</td>
<td>Wavelet Scalar Quantization (WSQ)</td>
</tr>
<tr>
<td>iso19794_4</td>
<td>Finger image data according to ISO 19794-4</td>
</tr>
<tr>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>--------------------------------------</td>
</tr>
<tr>
<td>iso19794_5</td>
<td>Face image data according to ISO 19794-5</td>
</tr>
</tbody>
</table>

Table 6.94 type.workflow.feedback.image.format - values

6.5.19.34.2. Elements

None.

6.5.19.34.3. XSD Definition

```xml
<xs:simpleType name="type.workflow.feedback.image.format">
  <xs:restriction base="xs:string">
    <xs:enumeration value="bmp"/>
    <xs:enumeration value="jpeg"/>
    <xs:enumeration value="jpeg2000"/>
    <xs:enumeration value="png"/>
    <xs:enumeration value="wsq"/>
    <xs:enumeration value="iso19794_4"/>
    <xs:enumeration value="iso19794_5"/>
  </xs:restriction>
</xs:simpleType>
```

6.5.19.35. type.workflow.dependencies

List of dependencies between elements of the workflow. The dependencies always refer to elements from the reading sequence or the feedback definition. Cyclic dependencies MUST be detected in `addWorkflow` and MUST prevent the workflow from loading.

6.5.19.35.1. Attributes

None.

6.5.19.35.2. Elements

The type.workflow.dependencies elements are given in Table 6.95.

<table>
<thead>
<tr>
<th>Element name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FinishToStart*</td>
<td><code>wf:type.workflow.dependencies.finishtostart</code></td>
</tr>
</tbody>
</table>

A dependency where an elements has to wait for another element.

Table 6.95 type.workflow.dependencies - elements

6.5.19.35.3. XSD Definition

```xml
<xs:complexType name="type.workflow.dependencies">
  <xs:choice minOccurs="0" maxOccurs="unbounded">
    <xs:element name="FinishToStart" type="wf:type.workflow.dependencies.finishtostart"/>
  </xs:choice>
</xs:complexType>
```

6.5.19.36. type.workflow.dependencies.finishtostart

Declares a sequential dependency. The processing of the first element MUST be finished before the processing of the dependent element is started.

6.5.19.36.1. Attributes

The type.workflow.dependencies.finishtostart attributes are given in Table 6.96.
### Table 6.96 type.workflow.dependencies.finishtostart - attributes

<table>
<thead>
<tr>
<th>Attribute name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id1*</td>
<td>xs:IDREF</td>
</tr>
<tr>
<td>Reference to a data or feedback element. Processing of this element MUST be finished before processing of id2 starts.</td>
<td></td>
</tr>
<tr>
<td>id2*</td>
<td>xs:IDREF</td>
</tr>
<tr>
<td>Reference to a data or feedback element. Processing of this element MUST NOT be started before processing of id1 finishes.</td>
<td></td>
</tr>
<tr>
<td>condition?</td>
<td>xs:IDREF</td>
</tr>
<tr>
<td>Reference to a condition. Controls whether this dependency is applicable. MUST be considered fullfilled if empty or attribute was omitted.</td>
<td></td>
</tr>
</tbody>
</table>

#### 6.5.19.36.2. Elements

None.

#### 6.5.19.36.3. XSD Definition

```xml
<xs:complexType name="type.workflow.dependencies.finishtostart">
  <xs:attribute name="id1" type="xs:IDREF" use="required"/>
  <xs:attribute name="id2" type="xs:IDREF" use="required"/>
  <xs:attribute name="condition" type="xs:IDREF"/>
</xs:complexType>
```

#### 6.5.19.37. type.workflow.extension

Allows access to implementation-specific workflow extensions. The server MUST consider known extensions during validation of the workflow definition. The server MUST reject unknown extensions with an appropriate WorkflowParserError message.

#### 6.5.19.37.1. Attributes

None.

#### 6.5.19.37.2. Elements

MAY contain a single element of any type from a different namespace than [http://trdoccheck.bsi.bund.de/hldc/workflow/2](http://trdoccheck.bsi.bund.de/hldc/workflow/2).

#### 6.5.19.37.3. XSD Definition

```xml
<xs:complexType name="type.workflow.extension">
  <xs:sequence>
    <xs:any namespace="#other" minOccurs="0" processContents="lax"/>
  </xs:sequence>
</xs:complexType>
```
7. Workflow feedback schema

Feedback requested by elements of type `type.workflow.feedback.xml` in the workflow definition is provided as an XML document in the `stringFeedback` element of `hldc:WorkflowFeedback`.

The XML schema for the XML feedback is located in the file `fb_v2.xsd` and is based on XSD 1.0.

7.1. Feedback document

XML document that contains feedback requested by elements of type `type.workflow.feedback.xml` in the workflow definition.

7.1.1. Root element

<table>
<thead>
<tr>
<th>Element name</th>
<th>Description</th>
</tr>
</thead>
</table>
| Feedback     | `wf:type.feedback`  
Root element of the feedback document. |

Table 7.1.

7.1.2. XSD Definition

```xml
<xs:element name="Feedback" type="wf:type.feedback" />
```

7.2. `type.feedback`

Root element of the XML feedback document. If an XPath expression was specified in the workflow definition of the relevant feedback request, the result is always returned in `XPathResult`. In all other cases, the feedback is returned in the corresponding element.

7.2.1. Attributes

<table>
<thead>
<tr>
<th>Attribute name</th>
<th>Description</th>
</tr>
</thead>
</table>
| schemaVersion  | `xs:decimal`  
Feedback schema version. MUST be “1”. |

Table 7.2.

7.2.2. Elements

type.feedback MUST contain only one of the following elements.

<table>
<thead>
<tr>
<th>Element name</th>
<th>Description</th>
</tr>
</thead>
</table>
| DG1Data      | `wf:type.feedback.dg1`  
Data from data group 1. |
| DG2Data      | `wf:type.feedback.dg2`  
Data from data group 2. |
<table>
<thead>
<tr>
<th>Element name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DG3Data</td>
<td>\textit{wf:type.feedback.dg3} Data from data group 3.</td>
</tr>
<tr>
<td>DG4Data</td>
<td>\textit{wf:type.feedback.dg4} Data from data group 4.</td>
</tr>
<tr>
<td>DG11Data</td>
<td>\textit{wf:type.feedback.dg11} Data from data group 11.</td>
</tr>
<tr>
<td>DG12Data</td>
<td>\textit{wf:type.feedback.dg12} Data from data group 12.</td>
</tr>
<tr>
<td>EIDDG17Data</td>
<td>\textit{wf:type.feedback.eid.placeofresidence} Data from eID data group 17.</td>
</tr>
<tr>
<td>DefectInfo</td>
<td>\textit{wf:type.feedback.defects} Defect information.</td>
</tr>
<tr>
<td>OpticalMRZData</td>
<td>\textit{wf:type.feedback.optmrz} Data from the optical MRZ.</td>
</tr>
<tr>
<td>SealVisa</td>
<td>\textit{wf:type.feedback.seal.visa} Data from the digital seal on a visa.</td>
</tr>
<tr>
<td>SealEmergencyTravelDocument</td>
<td>\textit{wf:type.feedback.seal.emergencytraveldocument} Data from the digital seal on an emergency travel document.</td>
</tr>
<tr>
<td>SealArrivalAttestationDocument</td>
<td>\textit{wf:type.feedback.seal.arrivalattestationdocument} Data from the digital seal of an arrival attestation document.</td>
</tr>
<tr>
<td>SealResidencePermit</td>
<td>\textit{wf:type.feedback.seal.residencepermit} Data from the digital seal on a residence permit.</td>
</tr>
<tr>
<td>SealResidencePermitSupplementarySheet</td>
<td>\textit{wf:type.feedback.seal.residencepermitsupplementarysheet} Data from the digital seal on a residence permit supplementay sheet.</td>
</tr>
<tr>
<td>SealAddressStickerGermanIdentityCard</td>
<td>\textit{wf:type.feedback.seal.addressstickergermanidentitycard} Data from the digital seal on a german identity card address sticker.</td>
</tr>
<tr>
<td>SealUnknown</td>
<td>\textit{wf:type.feedback.seal.unknown} Data from the digital seal of an unknown type.</td>
</tr>
<tr>
<td>ElectronicCheckResult</td>
<td>\textit{wf:type.feedback.checkresult} Result of an electronic check. Formatted according to the relevant document check namespace defined in [BSI TR-03135-1], e.g. \url{<a href="http://trdoccheck.bsi.bund.de/dce/4%7D">http://trdoccheck.bsi.bund.de/dce/4}</a>.</td>
</tr>
<tr>
<td>OpticalCheckResult</td>
<td>\textit{wf:type.feedback.checkresult} Result of an optical check. Formatted according to the relevant document check namespace defined in [BSI TR-03135-1], e.g. \url{<a href="http://trdoccheck.bsi.bund.de/dco/4%7D">http://trdoccheck.bsi.bund.de/dco/4}</a>.</td>
</tr>
<tr>
<td>SealCheckResult</td>
<td>\textit{wf:type.feedback.checkresult} Result of a seal check. Formatted according to the relevant document check namespace defined in [BSI TR-03135-1], e.g. \url{<a href="http://trdoccheck.bsi.bund.de/dcs/1%7D">http://trdoccheck.bsi.bund.de/dcs/1}</a>.</td>
</tr>
</tbody>
</table>
Table 7.3

<table>
<thead>
<tr>
<th>Element name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CombinedCheckResult</td>
<td>Result of a combined check. Formatted according to the relevant document check namespace defined in [BSI TR-03135-1], e.g. <a href="http://trdoccheck.bsi.bund.de/dcc/4">http://trdoccheck.bsi.bund.de/dcc/4</a>.</td>
</tr>
<tr>
<td>AggregatedDocumentCheckResult</td>
<td>Result of an aggregation of document check results.</td>
</tr>
<tr>
<td>XPathResult</td>
<td>Subset of XML nodes of the original feedback (datagroup or check result).</td>
</tr>
<tr>
<td>ExtensionFeedback</td>
<td>Implementation-specific feedback.</td>
</tr>
</tbody>
</table>

7.2.3. XSD Definition

```xml
<xs:complexType name="type.feedback">
  <xs:choice>
    <xs:element name="DG1Data" type="wf:type.feedback.dg1" />
    <xs:element name="DG2Data" type="wf:type.feedback.dg2" />
    <xs:element name="DG3Data" type="wf:type.feedback.dg3" />
    <xs:element name="DG4Data" type="wf:type.feedback.dg4" />
    <xs:element name="DG11Data" type="wf:type.feedback.dg11" />
    <xs:element name="DG12Data" type="wf:type.feedback.dg12" />
    <xs:element name="EIDDG17Data" type="wf:type.feedback.eid.placeofresidence" />
    <xs:element name="DefectInfo" type="wf:type.feedback.defects" />
    <xs:element name="OpticalMRZData" type="wf:type.feedback.optmrz" />
    <xs:element name="SealVisa" type="wf:type.feedback.seal.visa" />
    <xs:element name="SealEmergencyTravelDocument" type="wf:type.feedback.seal.emergencytraveldocument" />
    <xs:element name="SealArrivalAttestationDocument" type="wf:type.feedback.seal.arrivalattestationdocument" />
    <xs:element name="SealSocialInsuranceCard" type="wf:type.feedback.seal.socialinsurancecard" />
    <xs:element name="SealResidencePermit" type="wf:type.feedback.seal.residencepermit" />
    <xs:element name="SealResidencePermitSupplementarySheet" type="wf:type.feedback.seal.residencepermitsupplementarysheet" />
    <xs:element name="SealAddressStickerGermanIdentityCard" type="wf:type.feedback.seal.addressstickergermanidentitycard" />
    <xs:element name="SealUnknown" type="wf:type.feedback.seal.unknown" />
    <xs:element name="ElectronicCheckResult" type="wf:type.feedback.checkresult" />
    <xs:element name="OpticalCheckResult" type="wf:type.feedback.checkresult" />
    <xs:element name="CombinedCheckResult" type="wf:type.feedback.checkresult" />
    <xs:element name="CrossDocumentCheckResult" type="wf:type.feedback.checkresult" />
    <xs:element name="AggregatedDocumentCheckResult" type="wf:type.feedback.checkresult" />
    <xs:element name="XPathResult" type="wf:type.feedback.xpathresult" />
    <xs:element name="ExtensionFeedback" type="wf:type.feedback.extended" />
  </xs:choice>
  <xs:attribute name="schemaVersion" type="xs:decimal" use="required" />
</xs:complexType>
```

7.3. type.feedback.dg1

Contains the data from data group 1.
### 7.3.1. Attributes

None

### 7.3.2. Elements

<table>
<thead>
<tr>
<th>Element name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DocumentType</td>
<td>xs:string Type of the document.</td>
</tr>
<tr>
<td>Issuer</td>
<td>xs:string Issuer of the document.</td>
</tr>
<tr>
<td>GivenName</td>
<td>xs:string Given name of the document holder.</td>
</tr>
<tr>
<td>Surname</td>
<td>xs:string Surname of the document holder.</td>
</tr>
<tr>
<td>DocumentNumber</td>
<td>xs:string Document number (without check digit).</td>
</tr>
<tr>
<td>Nationality</td>
<td>xs:string Nationality of the document holder.</td>
</tr>
<tr>
<td>DateOfBirth</td>
<td>xs:string Date of birth of the document holder, encoded in the same way as in the MRZ (YYMMDD).</td>
</tr>
<tr>
<td>Sex</td>
<td>xs:string Sex of the document holder.</td>
</tr>
<tr>
<td>ExpiryDate</td>
<td>xs:string Expiry date of the document.</td>
</tr>
<tr>
<td>OptionalData</td>
<td>xs:string Optional data stored in the MRZ.</td>
</tr>
<tr>
<td>ChkDigitDocumentNumber</td>
<td>xs:string Check digit of the document number.</td>
</tr>
<tr>
<td>ChkDigitDateOfBirth</td>
<td>xs:string Check digit of the date of birth.</td>
</tr>
<tr>
<td>ChkDigitExpiryDate</td>
<td>xs:string Check digit of the expiry date.</td>
</tr>
<tr>
<td>ChkDigitOptionalData</td>
<td>xs:string Check digit for optional data.</td>
</tr>
<tr>
<td>ChkDigitComposite</td>
<td>xs:string Check digit of the MRZ.</td>
</tr>
<tr>
<td>IsoDateOfBirth?</td>
<td>type.feedback.string.date The full date of birth.</td>
</tr>
<tr>
<td>IsoExpiryDate?</td>
<td>type.feedback.string.date The full date of expiry.</td>
</tr>
</tbody>
</table>

Table 7.4.
7.3.3. XSD Definition

```xml
<xs:complexType name="type.feedback.dg1">
  <xs:sequence>
    <xs:element name="DocumentType" type="xs:string"/>
    <xs:element name="Issuer" type="xs:string"/>
    <xs:element name="GivenName" type="xs:string"/>
    <xs:element name="Surname" type="xs:string"/>
    <xs:element name="DocumentNumber" type="xs:string"/>
    <xs:element name="Nationality" type="xs:string"/>
    <xs:element name="DateOfBirth" type="xs:string"/>
    <xs:element name="Sex" type="xs:string"/>
    <xs:element name="ExpiryDate" type="xs:string"/>
    <xs:element name="OptionalData" type="xs:string"/>
    <xs:element name="ChkDgtDocumentNumber" type="xs:string"/>
    <xs:element name="ChkDgtDateOfBirth" type="xs:string"/>
    <xs:element name="ChkDgtExpiryDate" type="xs:string"/>
    <xs:element name="ChkDgtOptionalData" type="xs:string"/>
    <xs:element name="ChkDgtComposite" type="xs:string"/>
    <xs:element name="IsoDateOfBirth" type="wf:type.feedback.string.date" minOccurs="0"/>
    <xs:element name="IsoDateOfExpiry" type="wf:type.feedback.string.date" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
```

7.4. type.feedback.dg2

Contains a list of facial image templates.

7.4.1. Attributes

None

7.4.2. Elements

<table>
<thead>
<tr>
<th>Element name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Template*</td>
<td>wf:type.feedback.dg2.template</td>
</tr>
<tr>
<td></td>
<td>A facial image template.</td>
</tr>
</tbody>
</table>

Table 7.5

7.4.3. XSD Definition

```xml
<xs:complexType name="type.feedback.dg2">
  <xs:sequence>
    <xs:element name="Template" minOccurs="0" maxOccurs="unbounded" type="wf:type.feedback.dg2.template"/>
  </xs:sequence>
</xs:complexType>
```

7.5. type.feedback.dg2.template

Contains Base64-encoded image data of a single face. Derived from `xs:base64binary`.

7.5.1. Attributes

None
7.5.2. Elements

<table>
<thead>
<tr>
<th>Element name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Image*</td>
<td>wf:type.feedback.dg2.image</td>
</tr>
<tr>
<td></td>
<td>A facial image template.</td>
</tr>
</tbody>
</table>

Table 7.6.

7.5.3. XSD Definition

```xml
<xs:complexType name="type.feedback.dg2">
  <xs:sequence>
    <xs:element name="Template" minOccurs="0" maxOccurs="unbounded"
      type="wf:type.feedback.dg2.template" />
  </xs:sequence>
</xs:complexType>
```

7.6. type.feedback.dg2.image

Contains Base64-encoded image data of a single face. Derived from `xs:base64Binary`.

7.6.1. Attributes

<table>
<thead>
<tr>
<th>Attribute name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>width</td>
<td>xs:int</td>
</tr>
<tr>
<td>width</td>
<td>The image width.</td>
</tr>
<tr>
<td>height</td>
<td>xs:int</td>
</tr>
<tr>
<td>height</td>
<td>The image height.</td>
</tr>
<tr>
<td>imageType</td>
<td>xs:unsignedByte</td>
</tr>
<tr>
<td>imageType</td>
<td>The type of the face image according to ISO 19794-5. Can contain the following values:</td>
</tr>
<tr>
<td></td>
<td>• 0x00 – Basic</td>
</tr>
<tr>
<td></td>
<td>• 0x01 – Full Frontal</td>
</tr>
<tr>
<td></td>
<td>• 0x02 – Token Frontal</td>
</tr>
<tr>
<td></td>
<td>• 0x03–0xFF – Reserved / Not used in the scope of this interface</td>
</tr>
</tbody>
</table>

Table 7.7.

7.6.2. Elements

None

7.6.3. XSD Definition

```xml
<xs:complexType name="type.feedback.dg2.image">
  <xs:simpleContent base="xs:base64Binary">
    <xs:extension base="xs:base64Binary">
      <xs:attribute name="width" type="xs:int" use="required" />
      <xs:attribute name="height" type="xs:int" use="required" />
      <xs:attribute name="imageType" type="xs:unsignedByte" use="required" />
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>
```
7.7. type.feedback.dg3
Contains a list of fingerprint templates.

7.7.1. Attributes
None.

7.7.2. Elements

<table>
<thead>
<tr>
<th>Element name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Template*</td>
<td>wf:type.feedback.dg3.template</td>
</tr>
<tr>
<td></td>
<td>A fingerprint template.</td>
</tr>
</tbody>
</table>

Table 7.8.

7.7.3. XSD Definition

```xml
<xs:complexType name="type.feedback.dg3">
  <xs:sequence>
    <xs:element name="Template" minOccurs="0" maxOccurs="unbounded"
                 type="wf:type.feedback.dg3.template"/>
  </xs:sequence>
</xs:complexType>
```

7.8. type.feedback.dg3.template
Contains a list of fingerprint images.

7.8.1. Attributes
None.

7.8.2. Elements

<table>
<thead>
<tr>
<th>Element name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Image*</td>
<td>wf:type.feedback.dg3.image</td>
</tr>
<tr>
<td></td>
<td>A fingerprint image.</td>
</tr>
</tbody>
</table>

Table 7.9.

7.8.3. XSD Definition

```xml
<xs:complexType name="type.feedback.dg3.template">
  <xs:sequence>
    <xs:element name="Image" minOccurs="0" maxOccurs="unbounded"
                 type="wf:type.feedback.dg3.image"/>
  </xs:sequence>
</xs:complexType>
```

7.9. type.feedback.dg3.image
Contains Base64-encoded image data of a single fingerprint according to ISO 19794-4. Derived from xs:base64binary.
7.9.1. Attributes

<table>
<thead>
<tr>
<th>Attribute name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>width</td>
<td>xs:int</td>
</tr>
<tr>
<td></td>
<td>The image width.</td>
</tr>
<tr>
<td>height</td>
<td>xs:int</td>
</tr>
<tr>
<td></td>
<td>The image height.</td>
</tr>
<tr>
<td>fingerPos</td>
<td>type.feedback.dg3.image.fingerpos</td>
</tr>
<tr>
<td></td>
<td>Finger position code according to ISO 19794-4.</td>
</tr>
</tbody>
</table>

Table 7.10.

7.9.2. Elements

None.

7.9.3. XSD Definition

```xml
<xs:complexType name="type.feedback.dg3.image">
  <xs:simpleContent>
    <xs:extension base="xs:base64Binary">
      <xs:attribute name="width" type="xs:int" use="required" />
      <xs:attribute name="height" type="xs:int" use="required" />
      <xs:attribute name="fingerPos" type="xs:int" use="required" />
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>
```

7.10. type.feedback.dg3.image.fingerpos

Represents finger position codes according to ISO 19794-4 (Table 6).

7.10.1. Values

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Unknown</td>
</tr>
<tr>
<td>1</td>
<td>Right thumb</td>
</tr>
<tr>
<td>2</td>
<td>Right index finger</td>
</tr>
<tr>
<td>3</td>
<td>Right middle finger</td>
</tr>
<tr>
<td>4</td>
<td>Right ring finger</td>
</tr>
<tr>
<td>5</td>
<td>Right little finger</td>
</tr>
<tr>
<td>6</td>
<td>Left thumb</td>
</tr>
<tr>
<td>7</td>
<td>Left index finger</td>
</tr>
<tr>
<td>8</td>
<td>Left middle finger</td>
</tr>
<tr>
<td>9</td>
<td>Left ring finger</td>
</tr>
<tr>
<td>10</td>
<td>Left little finger</td>
</tr>
<tr>
<td>13</td>
<td>Plain right four fingers</td>
</tr>
<tr>
<td>14</td>
<td>Plain left four fingers</td>
</tr>
</tbody>
</table>
### 7.10.2. XSD Definition

```xml
<xs:simpleType name="type.feedback.dg3.image.fingerpos">
  <xs:union>
    <xs:simpleType>
      <xs:restriction base="xs:unsignedByte">
        <xs:minInclusive value="0"/>
        <xs:maxInclusive value="10"/>
      </xs:restriction>
    </xs:simpleType>
    <xs:simpleType>
      <xs:restriction base="xs:unsignedByte">
        <xs:minInclusive value="13"/>
        <xs:maxInclusive value="15"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:union>
</xs:simpleType>
```

### 7.11. type.feedback.dg4

Contains a list of iris templates.

#### 7.11.1. Attributes

None.

#### 7.11.2. Elements

<table>
<thead>
<tr>
<th>Element name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Template*</td>
<td>wf:type.feedback.dg4.template</td>
</tr>
<tr>
<td></td>
<td>An iris template.</td>
</tr>
</tbody>
</table>

| Table 7.12   |

#### 7.11.3. XSD Definition

```xml
<xs:complexType name="type.feedback.dg4">
  <xs:sequence>
    <xs:element name="Template" minOccurs="0" maxOccurs="unbounded"
      type="wf:type.feedback.dg4.template"/>
  </xs:sequence>
</xs:complexType>
```

### 7.12. type.feedback.dg4.template

Contains a list of iris images.

#### 7.12.1. Attributes

None.
7.12.2. Elements

<table>
<thead>
<tr>
<th>Element name</th>
<th>Description</th>
</tr>
</thead>
</table>
| Image*       | wf:type.feedback.dg4.image
|              | An iris image. |

Table 7.13

7.12.3. XSD Definition

```xml
<xs:complexType name="type.feedback.dg4.template">
  <xs:sequence>
    <xs:element name="Image" minOccurs="0" maxOccurs="unbounded" type="wf:type.feedback.dg4.image" />
  </xs:sequence>
</xs:complexType>
```

7.13. type.feedback.dg4.image

Contains Base64-encoded image data of a single iris. Derived from `xs:base64Binary`.

7.13.1. Attributes

<table>
<thead>
<tr>
<th>Attribute name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>width</td>
<td>xs:int</td>
</tr>
<tr>
<td>height</td>
<td>xs:int</td>
</tr>
</tbody>
</table>

The image width.

The image height.

Table 7.14

7.13.2. Elements

None.

7.13.3. XSD Definition

```xml
<xs:complexType name="type.feedback.dg4.image">
  <xs:simpleContent>
    <xs:extension base="xs:base64Binary">
      <xs:attribute name="width" type="xs:int" use="required"/>
      <xs:attribute name="height" type="xs:int" use="required"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>
```

7.14. type.feedback.dg11

Contains the XML feedback for datagroup 11.

7.14.1. Attributes

None.
7.14.2. Elements

<table>
<thead>
<tr>
<th>Element name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>The full name of the person.</td>
</tr>
<tr>
<td>PersonalNumber</td>
<td>Personal number.</td>
</tr>
<tr>
<td>FullDateOfBirth</td>
<td>The full date of birth including the century in format DD.MM.YYYY.</td>
</tr>
<tr>
<td>PlaceOfBirth</td>
<td>The place of birth.</td>
</tr>
<tr>
<td>PermanentAddress</td>
<td>The permanent address.</td>
</tr>
<tr>
<td>TelephoneNumber</td>
<td>The telephone number.</td>
</tr>
<tr>
<td>Profession</td>
<td>The profession.</td>
</tr>
<tr>
<td>Title</td>
<td>The title of the person.</td>
</tr>
<tr>
<td>PersonalSummary</td>
<td>Personal summary.</td>
</tr>
<tr>
<td>TDNumbers</td>
<td>Additional travel document numbers.</td>
</tr>
<tr>
<td>CustodyInformation</td>
<td>Custody information.</td>
</tr>
<tr>
<td>OtherNames</td>
<td>List of additional names.</td>
</tr>
<tr>
<td>ProofOfCitizenship?</td>
<td>Image of the proof of citizenship.</td>
</tr>
</tbody>
</table>

Table 7.15.

7.14.3. XSD Definition

```xml
<xs:complexType name="type.feedback.dg11">
  <xs:sequence>
    <xs:element name="Name" type="wf:type.feedback.name"/>
    <xs:element name="PersonalNumber" type="xs:string"/>
    <xs:element name="FullDateOfBirth" type="xs:string"/>
    <xs:element name="PlaceOfBirth" type="xs:string"/>
    <xs:element name="PermanentAddress" type="xs:string"/>
    <xs:element name="TelephoneNumber" type="xs:string"/>
    <xs:element name="Profession" type="xs:string"/>
    <xs:element name="Title" type="xs:string"/>
    <xs:element name="PersonalSummary" type="xs:string"/>
    <xs:element name="TDNumbers" type="xs:string"/>
    <xs:element name="CustodyInformation" type="xs:string"/>
    <xs:element name="OtherNames" type="wf:type.feedback.namelist"/>
    <xs:element name="ProofOfCitizenship" minOccurs="0" type="xs:base64Binary"/>
  </xs:sequence>
</xs:complexType>
```
7.15. type.feedback.dg12

Contains the XML feedback for data group 12.

7.15.1. Attributes

None.

7.15.2. Elements

<table>
<thead>
<tr>
<th>Element name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IssuingAuthority</td>
<td>xs:string</td>
</tr>
<tr>
<td>DateOfIssue</td>
<td>xs:string</td>
</tr>
<tr>
<td>Endorsements</td>
<td>xs:string</td>
</tr>
<tr>
<td>TaxExitRequirements</td>
<td>xs:string</td>
</tr>
<tr>
<td>PersTime</td>
<td>xs:string</td>
</tr>
<tr>
<td>SnrPersSystem</td>
<td>xs:string</td>
</tr>
<tr>
<td>OtherPeople</td>
<td>wf:type.feedback.namelist</td>
</tr>
<tr>
<td>FrontImage?</td>
<td>xs:base64Binary</td>
</tr>
<tr>
<td>RearImage?</td>
<td>xs:base64Binary</td>
</tr>
</tbody>
</table>

Table 7.16.

7.15.3. XSD Definition

```xml
<xs:complexType name="type.feedback.dg12">
  <xs:sequence>
    <xs:element name="IssuingAuthority" type="xs:string"/>
    <xs:element name="DateOfIssue" type="xs:string"/>
    <xs:element name="Endorsements" type="xs:string"/>
    <xs:element name="TaxExitRequirements" type="xs:string"/>
    <xs:element name="PersTime" type="xs:string"/>
    <xs:element name="SnrPersSystem" type="xs:string"/>
    <xs:element name="OtherPeople" type="wf:type.feedback.namelist"/>
    <xs:element name="FrontImage" minOccurs="0" type="xs:base64Binary"/>
    <xs:element name="RearImage" minOccurs="0" type="xs:base64Binary"/>
  </xs:sequence>
</xs:complexType>
```
7.16. type.feedback.optmrz

Contains optical MRZ data.

7.16.1. Attributes

None.

7.16.2. Elements

<table>
<thead>
<tr>
<th>Element name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DocumentType</td>
<td>xs:string Type of the document.</td>
</tr>
<tr>
<td>Issuer</td>
<td>xs:string Issuer of the travel document.</td>
</tr>
<tr>
<td>GivenName</td>
<td>xs:string Given name of the document holder.</td>
</tr>
<tr>
<td>Surname</td>
<td>xs:string Surname of the document holder.</td>
</tr>
<tr>
<td>DocumentNumber</td>
<td>xs:string Document number (without check digit).</td>
</tr>
<tr>
<td>Nationality</td>
<td>xs:string Nationality of the document holder.</td>
</tr>
<tr>
<td>DateOfBirth</td>
<td>xs:string Date of birth of the document holder, encoded in the same way as in the MRZ (YYMMDD).</td>
</tr>
<tr>
<td>Sex</td>
<td>xs:string Sex of the document holder.</td>
</tr>
<tr>
<td>ExpiryDate</td>
<td>xs:string Expiry date of the document.</td>
</tr>
<tr>
<td>OptionalData</td>
<td>xs:string Optional data stored in the MRZ.</td>
</tr>
<tr>
<td>ChkDigitDocumentNumber</td>
<td>xs:string Check digit of the document number.</td>
</tr>
<tr>
<td>ChkDigitDateOfBirth</td>
<td>xs:string Check digit of the date of birth.</td>
</tr>
<tr>
<td>ChkDigitExpiryDate</td>
<td>xs:string Check digit of the expiry date.</td>
</tr>
<tr>
<td>ChkDigitOptionalData</td>
<td>xs:string Check digit for optional data.</td>
</tr>
<tr>
<td>ChkDigitComposite</td>
<td>xs:string Check digit of the MRZ.</td>
</tr>
<tr>
<td>IsoDateOfBirth?</td>
<td>xs:string The full date of birth.</td>
</tr>
</tbody>
</table>
### Workflow feedback schema

<table>
<thead>
<tr>
<th>Element name</th>
<th>Description</th>
</tr>
</thead>
</table>
| IsoExpiryDate? | xs:string  
  The full date of expiry. |

Table 7.17.

### 7.16.3. XSD Definition

```xml
<xs:complexType name="type.feedback.optmrz">
  <xs:sequence>
    <xs:element name="DocumentType" type="xs:string"/>
    <xs:element name="Issuer" type="xs:string"/>
    <xs:element name="GivenName" type="xs:string"/>
    <xs:element name="Surname" type="xs:string"/>
    <xs:element name="Nationality" type="xs:string"/>
    <xs:element name="DateOfBirth" type="xs:string"/>
    <xs:element name="Sex" type="xs:string"/>
    <xs:element name="ExpiryDate" type="xs:string"/>
    <xs:element name="OptionalData" type="xs:string"/>
    <xs:element name="ChkDgtDocumentNumber" type="xs:string"/>
    <xs:element name="ChkDgtDateOfBirth" type="xs:string"/>
    <xs:element name="ChkDgtExpiryDate" type="xs:string"/>
    <xs:element name="ChkDgtOptionalData" type="xs:string"/>
    <xs:element name="ChkDgtComposite" type="xs:string"/>
    <xs:element name="IsoDateOfBirth" type="wf:type.feedback.string.date" minOccurs="0"/>
    <xs:element name="IsoDateOfExpiry" type="wf:type.feedback.string.date" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
```

### 7.17. Type feedback.eid.placeofresidence

Type for eID data group 17 place of residence.

#### 7.17.1. Attributes

None.

#### 7.17.2. Elements

<table>
<thead>
<tr>
<th>Element name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>StructuredPlace</td>
<td>wf:type.feedback.eid.placeofresidence .structuredplace</td>
</tr>
<tr>
<td>FreeTextPlace</td>
<td>xs:string</td>
</tr>
<tr>
<td>NoPlaceInfo</td>
<td>xs:string</td>
</tr>
</tbody>
</table>

Indicates that no place of residence is present.

Table 7.18.

#### 7.17.3. XSD Definition

```xml
<xs:complexType name="type.feedback.eid.placeofresidence">
  <xs:choice minOccurs="0" maxOccurs="unbounded">
    <xs:element name="StructuredPlace">
      <xs:complexType name="feedback.eid.placeofresidence.structuredplace"/>
    </xs:element>
    <xs:element name="FreeTextPlace" type="xs:string"/>
    <xs:element name="NoPlaceInfo"/>
  </xs:choice>
</xs:complexType>
```
7.18. `type.feedback.eid.placeofresidence.structuredplace`
Type for structured place of residence information.

7.18.1. Attributes
None.

7.18.2. Elements

<table>
<thead>
<tr>
<th>Element name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>City</td>
<td>xs:string&lt;br&gt;Contains a structured representation of the place of residence.</td>
</tr>
<tr>
<td>Country</td>
<td>xs:string&lt;br&gt;Contains the country of the structured place.</td>
</tr>
<tr>
<td>Street?</td>
<td>xs:string&lt;br&gt;Contains the street of the structured place.</td>
</tr>
<tr>
<td>State?</td>
<td>xs:string&lt;br&gt;Contains the state of the structured place.</td>
</tr>
<tr>
<td>ZipCode?</td>
<td>xs:string&lt;br&gt;Contains the zip code of the structured place.</td>
</tr>
</tbody>
</table>

Table 7.19.

7.18.3. XSD Definition

```xml
<xs:complexType name="type.feedback.eid.placeofresidence.structuredplace">
  <xs:sequence>
    <xs:element name="City" type="xs:string" />
    <xs:element name="Country" type="xs:string" />
    <xs:element name="Street" type="xs:string" minOccurs="0" />
    <xs:element name="State" type="xs:string" minOccurs="0" />
    <xs:element name="ZipCode" type="xs:string" minOccurs="0" />
  </xs:sequence>
</xs:complexType>
```

7.19. `type.feedback.defects`
Contains a list of known defects for the current document.

7.19.1. Attributes
None.

7.19.2. Elements

<table>
<thead>
<tr>
<th>Element name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Defect*</td>
<td>wf:type.feedback.defects.defect&lt;br&gt;A known defect for the current document.</td>
</tr>
</tbody>
</table>

Table 7.20.
7.19.3. XSD Definition

```xml
<xs:complexType name="type.feedback.defects">
  <xs:sequence>
    <xs:element name="Defect" minOccurs="0" maxOccurs="unbounded"
      type="wf:type.feedback.defects.defect"/>
  </xs:sequence>
</xs:complexType>
```

7.20. type.feedback.oid

Description of an Oid (ASN.1 object identifier).

7.20.1. Format restrictions

The content MUST match the specified regular expression pattern, for example “1.2.3.4”.

7.20.2. XSD Definition

```xml
<xs:complexType name="type.feedback.oid">
  <xs:restriction base="xs:string">
    <xs:pattern value="(\d+\.\d+)+"/>
  </xs:restriction>
</xs:complexType>
```

7.21. type.feedback.defects.defect

Description of a defect.

7.21.1. Attributes

None.

7.21.2. Elements

<table>
<thead>
<tr>
<th>Element name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oid</td>
<td><code>wf:type.feedback.oid</code></td>
</tr>
<tr>
<td>ParameterInfo?</td>
<td>The defect OID according to TR-03129-2.</td>
</tr>
<tr>
<td></td>
<td><code>xs:string</code></td>
</tr>
<tr>
<td></td>
<td>Additional information about the defect which must be interpreted in the</td>
</tr>
<tr>
<td></td>
<td>context of the defect type.</td>
</tr>
</tbody>
</table>

Table 7.21.

7.21.3. XSD Definition

```xml
<xs:complexType name="type.feedback.defects.defect">
  <xs:sequence>
    <xs:element name="Oid" type="wf:type.feedback.oid" />
    <xs:element name="ParameterInfo" minOccurs="0" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
```
7.22. type.feedbacks.name

Detailed name information.

7.22.1. Attributes
None.

7.22.2. Elements

<table>
<thead>
<tr>
<th>Element name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FullName</td>
<td>xs:string</td>
</tr>
<tr>
<td></td>
<td>The full name.</td>
</tr>
<tr>
<td>GivenName</td>
<td>xs:string</td>
</tr>
<tr>
<td></td>
<td>The given name.</td>
</tr>
<tr>
<td>Surname</td>
<td>xs:string</td>
</tr>
<tr>
<td></td>
<td>The surname.</td>
</tr>
</tbody>
</table>

Table 7.22.

7.22.3. XSD Definition

```xml
<xs:complexType name="type.feedback.name">
  <xs:sequence>
    <xs:element name="FullName" type="xs:string"/>
    <xs:element name="GivenName" type="xs:string"/>
    <xs:element name="Surname" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
```

7.23. type.feedback.namelist

List of names.

7.23.1. Attributes
None.

7.23.2. Elements

<table>
<thead>
<tr>
<th>Element name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name*</td>
<td>wf:type.feedback.name</td>
</tr>
<tr>
<td></td>
<td>The full name.</td>
</tr>
</tbody>
</table>

Table 7.23.

7.23.3. XSD Definition

```xml
<xs:complexType name="type.feedback.namelist">
  <xs:sequence>
    <xs:element name="Name" minOccurs="0" maxOccurs="unbounded" type="wf:type.feedback.name" />
  </xs:sequence>
</xs:complexType>
```
7.24. **type.feedback.checkresult**

Contains the results of an electronic, optical or combined check. Results are formatted according to the relevant document check schemas defined in [BSI TR-03135-1](http://trdoccheck.bsi.bund.de/hldc/workflow/1), e.g. `document_check_electronic_v2.xsd`.

### 7.24.1. Attributes

None.

### 7.24.2. Elements

MAY contain one element of any type from a different namespace than [http://trdoccheck.bsi.bund.de/hldc/workflow/1](http://trdoccheck.bsi.bund.de/hldc/workflow/1), which is the root element of the relevant document check schema.

### 7.24.3. XSD Definition

```xml
<xs:complexType name="type.feedback.extended">
  <xs:sequence>
    <xs:any namespace="#other" minOccurs="0" processContents="lax"/>
  </xs:sequence>
</xs:complexType>
```

7.25. **type.feedback.xpathresult**

Contains a subset of XML nodes of the original feedback (datagrid or check result) if an XPath expression is provided in the path element of `type.workflow.feedback.xml`.

### 7.25.1. Attributes

XPath-based feedback MAY contain any number of any attributes.

### 7.25.2. Elements

XPath-based feedback MAY contain any number of any elements.

### 7.25.3. XSD Definition

```xml
<xs:complexType name="type.feedback.xpathresult">
  <xs:sequence>
    <xs:any minOccurs="0" maxOccurs="unbounded" processContents="lax"/>
  </xs:sequence>
  <xs:anyAttribute processContents="lax"/>
</xs:complexType>
```

7.26. **type.feedback.seal.visa**

Contains the data of a digital seal on a visa.

### 7.26.1. Attributes

None.
7.26.2. Elements

<table>
<thead>
<tr>
<th>Element name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UnfoldedMRZ</td>
<td>wf:type.feedback.mrz.unfolded.shortened Truncated MRZ contained in the digital seal.</td>
</tr>
<tr>
<td>DurationOfStay</td>
<td>wf:type.feedback.seal.visa.durationofstay Duration of stay.</td>
</tr>
<tr>
<td>PassportNumber</td>
<td>wf:type.feedback.seal.passportnumber Number of the passport corresponding to the visa.</td>
</tr>
<tr>
<td>VisaType</td>
<td>xs:base64Binary Visa type encoded in the seal</td>
</tr>
<tr>
<td>NumberOfEntries</td>
<td>xs:unsignedByte Number of entries permitted using the visa document.</td>
</tr>
</tbody>
</table>

Table 7.24.

7.26.3. XSD Definition

```xml
<xs:complexType name="type.feedback.seal.visa">
  <xs:sequence>
    <xs:element name="UnfoldedMRZ" type="wf:type.feedback.mrz.unfolded.shortened"/>
    <xs:element name="DurationOfStay" type="wf:type.feedback.seal.visa.durationofstay"/>
    <xs:element name="PassportNumber" type="wf:type.feedback.seal.passportnumber"/>
    <xs:element name="VisaType" type="xs:base64Binary" minOccurs="0"/>
    <xs:element name="NumberOfEntries" type="xs:unsignedByte" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
```

7.27. type.feedback.seal.visa.durationofstay

Contains the duration permitted to stay according to the visa document.

7.27.1. Attributes

None.

7.27.2. Elements

<table>
<thead>
<tr>
<th>Element name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NumberOfDays</td>
<td>xs:unsignedByte Amount of days permitted to stay.</td>
</tr>
<tr>
<td>NumberOfMonths</td>
<td>xs:unsignedByte Amount of months permitted to stay.</td>
</tr>
<tr>
<td>NumberOfYears</td>
<td>xs:unsignedByte Amount of years permitted to stay.</td>
</tr>
</tbody>
</table>

Table 7.25.

7.27.3. XSD Definition

```xml
<xs:complexType name="type.feedback.seal.visa.durationofstay">
  <xs:sequence>
  </xs:sequence>
</xs:complexType>
```
7.28. **type.feedback.seal.emergencytraveldocument**

Contains the data of a digital seal on an emergency travel document.

**7.28.1. Attributes**

None.

**7.28.2. Elements**

<table>
<thead>
<tr>
<th>Element name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AdditionalFeatures</td>
<td>xs:string</td>
</tr>
<tr>
<td>UnfoldedMRZ</td>
<td>wf:type.feedback.mrz.unfolded</td>
</tr>
</tbody>
</table>

Features defined by the issuer.

Content of the MRZ contained in the emergency travel document.

**Table 7.26.**

**7.28.3. XSD Definition**

```xml
<xs:complexType name="type.feedback.seal.emergencytraveldocument">
  <xs:sequence>
    <xs:element name="AdditionalFeatures" type="xs:string"/>
    <xs:element name="UnfoldedMRZ" type="wf:type.feedback.mrz.unfolded"/>
  </xs:sequence>
</xs:complexType>
```

7.29. **type.feedback.seal.arrivalattestationdocument**

Contains the data of a digital seal on an arrival attestation document.

**7.29.1. Attributes**

None.

**7.29.2. Elements**

<table>
<thead>
<tr>
<th>Element name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AzrNumber</td>
<td>wf:type.feedback.seal.arrivalattestationdocumentazrnumber</td>
</tr>
<tr>
<td>UnfoldedMRZ</td>
<td>wf:type.feedback.mrz.unfolded</td>
</tr>
</tbody>
</table>

Number of the AZR.

Content of the MRZ contained in the arrival attestation document.

**Table 7.27.**

**7.29.3. XSD Definition**

```xml
<xs:complexType name="type.feedback.seal.arrivalattestationdocument">
  <xs:sequence>
  </xs:sequence>
</xs:complexType>
```
7.30. **type.feedback.seal.socialinsurancecard**

Contains the data of a digital seal on a social insurance card.

7.30.1. **Attribute**

None.

7.30.2. **Elements**

<table>
<thead>
<tr>
<th>Element name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SocialInsuranceNumber</td>
<td><code>wf:type.feedback.seal.socialinsurancecard.socialinsurancenumber</code></td>
</tr>
<tr>
<td>Surname</td>
<td><code>wf:type.feedback.seal.socialinsurancecard.surname</code></td>
</tr>
<tr>
<td>Firstname</td>
<td><code>wf:type.feedback.seal.socialinsurancecard.firstname</code></td>
</tr>
<tr>
<td>NameAtBirth</td>
<td><code>wf:type.feedback.seal.socialinsurancecard.nameatbirth</code></td>
</tr>
</tbody>
</table>

Table 7.28.

7.30.3. **XSD Definition**

```xml
<xs:complexType name="type.feedback.seal.socialinsurancecard">
  <xs:sequence>
    <xs:element name="SocialInsuranceNumber" type="wf:type.feedback.seal.socialinsurancecard.socialinsurancenumber"/>
    <xs:element name="Surname" type="wf:type.feedback.seal.socialinsurancecard.surname"/>
    <xs:element name="Firstname" type="wf:type.feedback.seal.socialinsurancecard.firstname"/>
    <xs:element name="NameAtBirth" type="wf:type.feedback.seal.socialinsurancecard.nameatbirth" minOccurs="0"/>
  </xs:sequence>
</xs:complexType>
```

7.31. **type.feedback.seal.residencepermit**

Contains the data of a digital seal on a residence permit.

7.31.1. **Attribute**

None.

7.31.2. **Elements**

<table>
<thead>
<tr>
<th>Element name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PassportNumber</td>
<td><code>wf:type.feedback.sealpassportnumber</code></td>
</tr>
</tbody>
</table>

Number of the corresponding passport.
### 7.31.3. XSD Definition

```xml
<xs:complexType name="type.feedback.seal.residencepermit">
  <xs:sequence>
    <xs:element name="PassportNumber" type="wf:type.feedback.seal.passportnumber"/>
    <xs:element name="UnfoldedMRZ" type="wf:type.feedback.mrz.unfolded"/>
  </xs:sequence>
</xs:complexType>
```

**7.32. `type.feedback.seal.residencepermitsupplementarysheet`**

Contains the data of a residence permit supplementary sheet.

#### 7.32.1. Attribute

None.

#### 7.32.2. Elements

<table>
<thead>
<tr>
<th>Element name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NumberOfSupplementarySheet</td>
<td><code>wf:type.feedback.seal.residencepermitsupplementarysheet</code></td>
</tr>
<tr>
<td>UnfoldedMRZ</td>
<td><code>wf:type.feedback.mrz.unfolded</code></td>
</tr>
</tbody>
</table>

Number of the supplementary sheet.

MRZ contained in the document.

#### 7.32.3. XSD Definition

```xml
<xs:complexType name="type.feedback.seal.residencepermitsupplementarysheet">
  <xs:sequence>
    <xs:element name="NumberOfSupplementarySheet" type="wf:type.feedback.seal.residencepermitsupplementarysheet.numberofsupplementarysheet"/>
    <xs:element name="UnfoldedMRZ" type="wf:type.feedback.mrz.unfolded"/>
  </xs:sequence>
</xs:complexType>
```

**7.33. `type.feedback.seal.addressstickergermanidentitycard`**

Contains the content of an address sticker of a german identity card.

#### 7.33.1. Attribute

None.
7.33.2. Elements

<table>
<thead>
<tr>
<th>Element name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DocumentNumber</td>
<td>xs:string Number of the document.</td>
</tr>
<tr>
<td>ResidentialAddress</td>
<td>xs:string Residential address.</td>
</tr>
</tbody>
</table>

Table 7.31.

7.33.3. XSD Definition

```
<xs:complexType name="type.feedback.seal.addressstickergermanidentitycard">
  <xs:sequence>
    <xs:element name="DocumentNumber" type="xs:string"/>
    <xs:element name="OfficialMunicipalityCodeNumber" type="xs:string"/>
    <xs:element name="ResidentialAddress" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
```

7.34. type.feedback.seal.unknown

Contains the content of an unknown seal type.

7.34.1. Attribute

None.

7.34.2. Elements

<table>
<thead>
<tr>
<th>Element name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BinaryContent</td>
<td>xs:base64Binary Digita seal content in an unsupported format.</td>
</tr>
</tbody>
</table>

Table 7.32.

7.34.3. XSD Definition

```
<xs:complexType name="type.feedback.seal.unknown">
  <xs:sequence>
    <xs:element name="BinaryContent" type="xs:base64Binary"/>
  </xs:sequence>
</xs:complexType>
```

7.35. type.feedback.mrz.unfolded

Contains an unfolded MRZ.

7.35.1. Attributes

None.
### 7.35.2. Elements

<table>
<thead>
<tr>
<th>Element name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DocumentType</td>
<td>The type of the document.</td>
</tr>
<tr>
<td>Issuer</td>
<td>The issuer of the document.</td>
</tr>
<tr>
<td>GivenName</td>
<td>The given name of the document holder.</td>
</tr>
<tr>
<td>Surname</td>
<td>The surname of the document holder.</td>
</tr>
<tr>
<td>DocumentNumber</td>
<td>The number of the document.</td>
</tr>
<tr>
<td>Nationality</td>
<td>The nationality of the document holder.</td>
</tr>
<tr>
<td>DateOfBirth</td>
<td>The date of birth of the document holder.</td>
</tr>
<tr>
<td>Sex</td>
<td>The sex of the document holder.</td>
</tr>
<tr>
<td>ExpiryDate</td>
<td>The expiry date of the document.</td>
</tr>
<tr>
<td>OptionalData</td>
<td>Optional data in the document.</td>
</tr>
<tr>
<td>ChkDgtDocumentNumber</td>
<td>The check digit of the document number.</td>
</tr>
<tr>
<td>ChkDgtDateOfBirth</td>
<td>The check digit of the date of birth number.</td>
</tr>
<tr>
<td>ChkDgtExpiryDate</td>
<td>The check digit of the expiry date.</td>
</tr>
<tr>
<td>ChkDgtOptionalData</td>
<td>The check digit of the optional data.</td>
</tr>
<tr>
<td>ChkDgtComposite</td>
<td>The composite check digit.</td>
</tr>
<tr>
<td>IsoDateOfBirth</td>
<td>The date of birth in ISO format.</td>
</tr>
<tr>
<td>IsoDateOfExpiry</td>
<td>The date of expiry in ISO format.</td>
</tr>
</tbody>
</table>

Table 7.33.

### 7.35.3. XSD Definition

```xml
<xs:complexType name="type.feedback.mrz.unfolded">
```

Federal Office for Information Security
7.36. type.feedback.mrz.unfolded.shortened

Contains an shortened unfolded MRZ, i.e. without optional data and optional data check digit.

7.36.1. Attributes

None.

7.36.2. Elements

<table>
<thead>
<tr>
<th>Element name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DocumentType</td>
<td>xs:string</td>
</tr>
<tr>
<td>Issuer</td>
<td>xs:string</td>
</tr>
<tr>
<td>GivenName</td>
<td>xs:string</td>
</tr>
<tr>
<td>Surname</td>
<td>xs:string</td>
</tr>
<tr>
<td>DocumentNumber</td>
<td>xs:string</td>
</tr>
<tr>
<td>Nationality</td>
<td>xs:string</td>
</tr>
<tr>
<td>DateOfBirth</td>
<td>xs:string</td>
</tr>
<tr>
<td>Sex</td>
<td>xs:string</td>
</tr>
<tr>
<td>ExpiryDate</td>
<td>xs:string</td>
</tr>
<tr>
<td>OptionalData</td>
<td>xs:string</td>
</tr>
<tr>
<td>ChkDgtDocumentNumber</td>
<td>xs:string</td>
</tr>
<tr>
<td>ChkDgtDateOfBirth</td>
<td>xs:string</td>
</tr>
<tr>
<td>ChkDgtExpiryDate</td>
<td>xs:string</td>
</tr>
<tr>
<td>ChkDgtOptionalData</td>
<td>xs:string</td>
</tr>
<tr>
<td>ChkDgtComposite</td>
<td>xs:string</td>
</tr>
<tr>
<td>IsoDateOfBirth</td>
<td>wf:type.feedback.string.date</td>
</tr>
<tr>
<td>IsoDateOfExpiry</td>
<td>wf:type.feedback.string.date</td>
</tr>
<tr>
<td>Element name</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>ChkDgtDocumentNumber</td>
<td>xs:string</td>
</tr>
<tr>
<td></td>
<td>The check digit of the document number.</td>
</tr>
<tr>
<td>ChkDgtDateOfBirth</td>
<td>xs:string</td>
</tr>
<tr>
<td></td>
<td>The check digit of the date of birth number.</td>
</tr>
<tr>
<td>ChkDgtExpiryDate</td>
<td>xs:string</td>
</tr>
<tr>
<td></td>
<td>The check digit of the expiry date.</td>
</tr>
<tr>
<td>ChkDgtComposite</td>
<td>xs:string</td>
</tr>
<tr>
<td></td>
<td>The composite check digit.</td>
</tr>
<tr>
<td>IsoDateOfBirth</td>
<td>wf:type.feedback.string.date</td>
</tr>
<tr>
<td></td>
<td>The date of birth in ISO format.</td>
</tr>
<tr>
<td>IsoDateOfExpiry</td>
<td>wf:type.feedback.string.date</td>
</tr>
<tr>
<td></td>
<td>The date of expiry in ISO format.</td>
</tr>
</tbody>
</table>

Table 7.34.

### 7.36.3. XSD Definition

```xml
<xs:complexType name="type.feedback.mrz.unfolded.shortened">
  <xs:sequence>
    <xs:element name="DocumentType" type="xs:string" />
    <xs:element name="Issuer" type="xs:string" />
    <xs:element name="GivenName" type="xs:string" />
    <xs:element name="Surname" type="xs:string" />
    <xs:element name="DocumentNumber" type="xs:string" />
    <xs:element name="Nationality" type="xs:string" />
    <xs:element name="DateOfBirth" type="xs:string" />
    <xs:element name="Sex" type="xs:string" />
    <xs:element name="ExpiryDate" type="xs:string" />
    <xs:element name="ChkDgtDocumentNumber" type="xs:string" />
    <xs:element name="ChkDgtDateOfBirth" type="xs:string" />
    <xs:element name="ChkDgtExpiryDate" type="xs:string" />
    <xs:element name="ChkDgtComposite" type="xs:string" />
    <xs:element name="IsoDateOfBirth" type="wf:type.feedback.string.date" minOccurs="0" />
    <xs:element name="IsoDateOfExpiry" type="wf:type.feedback.string.date" minOccurs="0" />
  </xs:sequence>
</xs:complexType>
```

### 7.37. type.feedback.string.date

Represents a date string with full year information. Missing parts shall be encoded with "00", e.g. "1990-10-00" or "2011-00-00".

#### 7.37.1. Values

The string SHALL conform to the pattern `\d{4}-\d{2}-\d{2}`.

#### 7.37.2. XSD Definition

```xml
<xs:simpleType name="type.feedback.string.date">
  <xs:restriction base="xs:string">
    <xs:pattern value="\d{4}-\d{2}-\d{2}" />
  </xs:restriction>
</xs:simpleType>
```
7.38. type.feedback.extended

Allows the server to return implementation-specific feedback. MUST only be used if the workflow definition contains corresponding requests for custom feedback as part of an implementation-specific workflow extension configuration.

7.38.1. Attributes

None.

7.38.2. Elements

Implementation-specific feedback MAY contain one element of any type from a different namespace than [http://trdoccheck.bsi.bund.de/hldc/workflow/1](http://trdoccheck.bsi.bund.de/hldc/workflow/1).

7.38.3. XSD Definition

```xml
<xs:complexType name="type.feedback.extended">
  <xs:sequence>
    <xs:any namespace="#other" minOccurs="0" processContents="lax"/>
  </xs:sequence>
</xs:complexType>
```

7.39. Workflow extensions

The workflow and feedback specification of the high-level document check allow for implementation-specific extensions. This section specifies the constraints for such extensions.

7.39.1. Workflow definition

1. All server implementations MUST correctly validate all workflow definitions that conform to this document.
2. All implementation-specific extensions MUST be placed below the `hldc#wf:Extension` node. To prevent conflicts between multiple extension implementations, only one extension is allowed per workflow.
3. A server MUST validate known extensions. Any errors MAY be reported with a regular validation error message as part of the `WorkflowParserError`.
4. A server MUST reject unknown extensions to prevent incomplete or incorrect interpretation of the workflow definition. The `WorkflowParserError` MUST explicitly state that the extension was rejected.
5. Extension elements MAY refer to IDs in the regular workflow definition and vice versa. This allows extensions that are transparent to the client (e.g. by adding a new type of condition).

7.39.2. Feedback

1. If the workflow extension does not define custom feedback elements, the workflow and its feedback MUST be processable by all clients that conform to this document without modification of the client.
2. All XML-based feedback MUST be validatable by all conformant clients.
3. All feedback requests that conform to this Technical Guideline MUST be answered by corresponding conformant feedback. Custom feedback MAY only be used for implementation-specific feedback requests.
4. Extensions MAY provide feedback as
• modifications of regular feedback
• text-based feedback in the stringFeedback field of WorkflowFeedback
• binary feedback in the binaryFeedback field of WorkflowFeedback
• Custom XML-based feedback using the wf:ExtensionFeedback node
8. Transaction management

The Transaction Management API provides functions for transaction-based logging of document checks in compliance to [BSI TR-03135-1].

The definitions of the Transaction Management API are provided in t1_v2.wsdl.

8.1. Namespace

All elements that are defined in this chapter are member of the namespace [http://trdoccheck.bsi.bund.de/tl/wsdl/2], which is aliased by t1.

8.2. Data types

In addition to simple XSD types, the SOAP interface uses custom data types, which are described in the following.

8.2.1. ExternalKey

Represents a name-value combination which will be included in the [BSI TR-03135] XML data.

8.2.1.1. Attributes

None.

8.2.1.2. Elements

<table>
<thead>
<tr>
<th>Element name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>xs:string</td>
</tr>
<tr>
<td>value</td>
<td>xs:string</td>
</tr>
</tbody>
</table>

Name of the external key.

Value of the external key.

Table 8.1

8.2.1.3. WSDL Definition

```xml
<complexType name="ExternalKey">
  <sequence>
    <element name="name" type="xs:string" />
    <element name="value" type="xs:string" />
  </sequence>
</complexType>
```

8.2.2. LoggingProvider

Represents a [BSI TR-03135]-compliant logging provider which saves [BSI TR-03135] XML data persistently.

8.2.2.1. Attributes

None.
8.2.2.2. Elements

<table>
<thead>
<tr>
<th>Element name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>xs:string</td>
</tr>
<tr>
<td></td>
<td>Name of the logging provider.</td>
</tr>
<tr>
<td>type</td>
<td>xs:string</td>
</tr>
<tr>
<td></td>
<td>The type of the logging provider (can be empty).</td>
</tr>
</tbody>
</table>

Table 8.2.

8.2.2.3. WSDL Definition

```xml
<complexType name="LoggingProvider">
  <sequence>
    <element name="name" type="xs:string"/>
    <element name="type" type="xs:string"/>
  </sequence>
</complexType>
```

8.2.3. LogType


8.2.3.1. Values

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>doc-check</td>
<td>Document check with [BSI TR-03135]-compliant XML data.</td>
</tr>
<tr>
<td>bio-check</td>
<td>Biometric check with [BSI TR-03135]-compliant XML data.</td>
</tr>
<tr>
<td>background-check</td>
<td>Background system check with arbitrary XML data.</td>
</tr>
<tr>
<td>app-specific-check</td>
<td>Application specific check with arbitrary XML data.</td>
</tr>
</tbody>
</table>

Table 8.3.

8.2.3.2. WSDL Definition

```xml
<simpleType name="LogType">
  <restriction base="xs:string">
    <enumeration value="doc-check"/>
    <enumeration value="bio-check"/>
    <enumeration value="background-check"/>
    <enumeration value="app-specific-check"/>
  </restriction>
</simpleType>
```

8.2.4. UUID

The definition of tl:UUID is identical to hldc:UUID except for the namespace. Please refer to Section 6.2.2 for the definition.

8.3. Fault types

This section specifies the SOAP faults that are specific to this SOAP API. No fault has any attributes.
8.3.1.
Returned by methods that prevent further modification of a transaction (read-only) if the transaction is currently written to.

8.3.1.1. WSDL Definition

```xml
<complexType name="In Progress">
  <sequence>
  </sequence>
</complexType>
```

8.3.2. InvalidLoggingProvider
Returned if an invalid logging provider was requested or if no default logging provider is available.

8.3.2.1. WSDL Definition

```xml
<complexType name="InvalidLoggingProvider">
  <sequence>
  </sequence>
</complexType>
```

8.3.3. InvalidLoggingParameter
Returned if the logging provider cannot interpret the provider-specific parameter.

8.3.3.1. WSDL Definition

```xml
<complexType name="InvalidLoggingParameter">
  <sequence>
  </sequence>
</complexType>
```

8.3.4. InvalidTransactionId
Returned if a transaction ID does not exist on the server. The ID is either invalid or has expired due to a call to endTransaction or limited ressources on the server.

8.3.4.1. Elements

<table>
<thead>
<tr>
<th>Element name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id?</td>
<td>tl:UUID</td>
</tr>
<tr>
<td></td>
<td>The rejected ID. Only present in case of ambiguity.</td>
</tr>
</tbody>
</table>

Table 8.4.

8.3.4.2. WSDL Definition

```xml
<complexType name="InvalidTransactionId">
  <sequence>
    <element name="id" type="tl:UUID" minOccurs="0"/>
  </sequence>
</complexType>
```
8.3.5. LoggingFailed
Returned if the server could not complete the logging procedure.

8.3.5.1. WSDL Definition

```
<complexType name="InvalidTransactionId">
  <sequence>
    <element name="id" type="tl:UUID" minOccurs="0" />
  </sequence>
</complexType>
```

8.3.6. LoggingProfileNotFound
Returned if the requested logging profile does not exist on the server.

8.3.6.1. WSDL Definition

```
<complexType name="LoggingProfileNotFound">
  <sequence>
  </sequence>
</complexType>
```

8.3.7. LogParserError
Returned if the server could not parse submitted log XML or logging profile XSLT.

8.3.7.1. WSDL Definition

```
<complexType name="LogParserError">
  <sequence>
  </sequence>
</complexType>
```

8.3.8. ReadOnly
Returned if the transaction is read-only and must not be modified.

8.3.8.1. WSDL Definition

```
<complexType name="ReadOnly">
  <sequence>
  </sequence>
</complexType>
```

8.4. Operations

8.4.1. addLogData
Allows the client application to add custom log data to a transaction.

8.4.1.1. Request elements

<table>
<thead>
<tr>
<th>Element name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>transactionID</td>
<td>t1:UUID</td>
</tr>
<tr>
<td></td>
<td>The transaction ID to which the data should be copied.</td>
</tr>
</tbody>
</table>
### Transaction management

<table>
<thead>
<tr>
<th>Element name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>logType</td>
<td>t1:LogType</td>
</tr>
<tr>
<td></td>
<td>Specifies the type of the log data.</td>
</tr>
<tr>
<td>content</td>
<td>xs:string</td>
</tr>
<tr>
<td></td>
<td>The xml-formatted log data.</td>
</tr>
</tbody>
</table>

**Table 8.5.**

#### 8.4.1.2. Response elements

None.

#### 8.4.1.3. Faults

<table>
<thead>
<tr>
<th>Fault type</th>
<th>Cause</th>
</tr>
</thead>
<tbody>
<tr>
<td>t1:InvalidTransactionId</td>
<td>Value of transactionID is invalid or has expired.</td>
</tr>
<tr>
<td>t1:LogParserError</td>
<td>Value of content could not be parsed or validated.</td>
</tr>
<tr>
<td>t1:ReadOnly</td>
<td>The transaction is write-protected and must not be modified.</td>
</tr>
</tbody>
</table>

**Table 8.6.**

#### 8.4.1.4. WSDL Definition

```xml
<element name="addLogData">
  <complexType>
    <sequence>
      <element name="transactionID" type="t1:UUID" />
      <element name="logType" type="t1:LogType" />
      <element name="content" type="xs:string" />
    </sequence>
  </complexType>
</element>
```

#### 8.4.2. addLoggingProfile

Transfers a new logging profile definition to the server. The profile must be a valid XSL transformation that operates on a transaction log according to Part 1 of this Technical Guideline. If an identically named workflow exists on the server, it MUST be replaced with the newly submitted definitions.

#### 8.4.2.1. Request elements

<table>
<thead>
<tr>
<th>Element name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>xs:string</td>
</tr>
<tr>
<td></td>
<td>Reference to the profile for use with applicable SOAP calls.</td>
</tr>
<tr>
<td>profile</td>
<td>xs:base64Binary</td>
</tr>
<tr>
<td></td>
<td>The base64-encoded XSLT document.</td>
</tr>
</tbody>
</table>

**Table 8.7.**
8.4.2.2. **Response elements**

None.

8.4.2.3. **Faults**

<table>
<thead>
<tr>
<th>Fault type</th>
<th>Cause</th>
</tr>
</thead>
<tbody>
<tr>
<td>tl:LogParserError</td>
<td>An error occurred while validating the submitted logging profile.</td>
</tr>
</tbody>
</table>

Table 8.8

8.4.2.4. **WSDL Definition**

```xml
<!-- operation request element -->
<element name="addLoggingProfile">
  <complexType>
    <sequence>
      <element name="name" type="xs:string"/>
      <element name="profile" type="xs:base64Binary"/>
    </sequence>
  </complexType>
</element>

<!-- operation response element -->
<element name="addLoggingProfileResponse">
  <complexType>
    <sequence>
    </sequence>
  </complexType>
</element>
```

8.4.3. **beginTransaction**

Opens a [BSI TR-03135] transaction and returns a new transaction ID.

8.4.3.1. **Request elements**

<table>
<thead>
<tr>
<th>Element name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ctNamespace</td>
<td>xs:string The “Check Transaction” namespace for the logging of the [BSI TR-03135] data. If empty, [<a href="http://trdoccheck.bsi.bund.de/ct/4%5C">http://trdoccheck.bsi.bund.de/ct/4\</a>] is assumed.</td>
</tr>
<tr>
<td>schemaVersion</td>
<td>xs:string The [BSI TR-03135] schema version number. If empty, version 1 is assumed.</td>
</tr>
<tr>
<td>location</td>
<td>xs:string The location of the inspection system. Copied into the [BSI TR-03135] XML data.</td>
</tr>
<tr>
<td>operationalEnvironment</td>
<td>xs:string The operational environment which is copied into the TR#03135 XML data. Examples: entry, exit, domestic, training, mobile, undefined.</td>
</tr>
<tr>
<td>externalKeys*</td>
<td>xs:string External keys to be copied into the [BSI TR-03135] XML data.</td>
</tr>
</tbody>
</table>

Table 8.9
## 8.4.3.2. Response elements

<table>
<thead>
<tr>
<th>Element name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>transactionID</td>
<td>tl:UUID</td>
</tr>
</tbody>
</table>

Table 8.10.

## 8.4.3.3. Faults

None.

## 8.4.3.4. WSDL Definition

```xml
<!-- operation request element -->
<element name="beginTransaction">
  <complexType>
    <sequence>
      <element name="ctNamespace" type="xs:string" />
      <element name="schemaVersion" type="xs:string" />
      <element name="location" type="xs:string" />
      <element name="operationalEnvironment" type="xs:string" />
      <element name="externalKeys" type="tl:ExternalKey"
        minOccurs="0" maxOccurs="unbounded"/>
    </sequence>
  </complexType>
</element>

<!-- operation response element -->
<element name="beginTransactionResponse">
  <complexType>
    <sequence>
      <element name="transactionID" type="tl:UUID" />
    </sequence>
  </complexType>
</element>
```

## 8.4.4. endTransaction

Closes a transaction. Invalidates the corresponding transaction ID.

### 8.4.4.1. Request elements

<table>
<thead>
<tr>
<th>Element name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>transactionID</td>
<td>tl:UUID</td>
</tr>
</tbody>
</table>

Table 8.11.

### 8.4.4.2. Response elements

None.

### 8.4.4.3. Faults

<table>
<thead>
<tr>
<th>Fault type</th>
<th>Cause</th>
</tr>
</thead>
<tbody>
<tr>
<td>tl:InvalidTransactionId</td>
<td>Value of transactionID is invalid or has expired.</td>
</tr>
</tbody>
</table>

Table 8.12.
8.4.4.4. WSDL Definition

```xml
<element name="endTransaction">
  <complexType>
    <sequence>
      <element name="transactionID" type="tl:UUID"/>
    </sequence>
  </complexType>
</element>
```

8.4.5. getAllLoggingProviders

Returns a list of all available logging providers. This function is OPTIONAL.

8.4.5.1. Request elements

None.

8.4.5.2. Response elements

<table>
<thead>
<tr>
<th>Element name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>loggingProvider*</td>
<td>t1:LoggingProvider loggingProvider*</td>
</tr>
</tbody>
</table>

Available logging providers.

Table 8.13.

8.4.5.3. Faults

None.

8.4.5.4. WSDL Definition

```xml
<element name="getAllLoggingProviders">
  <complexType>
    <sequence>
      <element name="loggingProvider" type="t1:LoggingProvider" minOccurs="0" maxOccurs="unbounded"/>
    </sequence>
  </complexType>
</element>
```

8.4.6. getTransactionXML

Returns the [BSI TR-03135]-compliant XML data of the corresponding transaction. If called, MUST be called between beginTransaction and endTransaction.
If the transaction is currently being modified, e.g., by the document check process, the server MAY fail the call with `InProgress` fault.

On success, the transaction MUST be flagged as read-only to prevent further modification, if the `keepWritable` is not present or has the value `false`, otherwise – if the `keepWritable` attribute has the value `true`, the transaction MUST be kept open for further writes.

### 8.4.6.1. Request elements

<table>
<thead>
<tr>
<th>Element name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>transactionID</code></td>
<td>A valid transaction ID.</td>
</tr>
<tr>
<td><code>errorCode</code></td>
<td>Application-defined error code which is copied to the [BSI TR-03135]-XML. If there was no error, the <code>errorCode</code> MUST be set to 0.</td>
</tr>
<tr>
<td><code>loggingProfile</code></td>
<td>Name of the logging profile.</td>
</tr>
<tr>
<td><code>keepWritable</code></td>
<td>Whether the transaction XML is kept open for further writes.</td>
</tr>
</tbody>
</table>

Table 8.14.

### 8.4.6.2. Response elements

<table>
<thead>
<tr>
<th>Element name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>xml</code></td>
<td>The [BSI TR-03135]-compliant XML data of the specified transaction.</td>
</tr>
</tbody>
</table>

Table 8.15.

### 8.4.6.3. Faults

<table>
<thead>
<tr>
<th>Fault type</th>
<th>Cause</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>tl:InProgress</code></td>
<td>The transaction is currently being written to and cannot be write-protected.</td>
</tr>
<tr>
<td><code>tl:InvalidTransactionId</code></td>
<td>Value of transactionID is invalid or has expired.</td>
</tr>
<tr>
<td><code>tl:LoggingProfileNotFound</code></td>
<td>The requested logging profile <code>loggingProfile</code> does not exist on the server.</td>
</tr>
</tbody>
</table>

Table 8.16.

### 8.4.6.4. WSDL Definition

```xml
<!-- operation request element -->
<element name="getTransactionXML">
  <complexType>
    <sequence>
      <element name="transactionID" type="tl:UUID" />
      <element name="errorCode" type="xs:int" />
      <element name="loggingProfile" type="xs:string" />
      <element name="keepWritable" type="xs:boolean" minOccurs="0" />
    </sequence>
  </complexType>
</element>
<!-- operation response element -->
<element name="getTransactionXMLResponse">
...</element>
```
### 8.4.7. mergeTransaction

Merges contents of the source transaction into the target transaction and ends the source transaction afterwards.

#### 8.4.7.1. Request elements

<table>
<thead>
<tr>
<th>Element name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>targetID</td>
<td>t1:UUID Transaction ID of the target transaction.</td>
</tr>
<tr>
<td>sourceID</td>
<td>t1:UUID Transaction ID of the source transaction.</td>
</tr>
</tbody>
</table>

Table 8.17.

#### 8.4.7.2. Response elements

None.

#### 8.4.7.3. Faults

<table>
<thead>
<tr>
<th>Fault type</th>
<th>Cause</th>
</tr>
</thead>
<tbody>
<tr>
<td>t1::InProgress</td>
<td>The transaction sourceID is currently being written to and cannot be write-protected.</td>
</tr>
<tr>
<td>t1::InvalidTransactionId</td>
<td>Either the value of targetID or sourceID is invalid or has expired</td>
</tr>
<tr>
<td>t1::ReadOnly</td>
<td>The transaction targetID is write-protected and must not be modified.</td>
</tr>
</tbody>
</table>

Table 8.18.

#### 8.4.7.4. WSDL Definition

```xml
<complexType>
  <sequence>
    <element name="xml" type="xs:string" />
  </sequence>
</complexType>
```

#### 8.4.8. removeLoggingProfile

Removes a logging profile which was previously loaded with addLoggingProfile
8.4.8.1. Request elements

<table>
<thead>
<tr>
<th>Element name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>profileName</td>
<td>tl:string</td>
</tr>
<tr>
<td></td>
<td>The name of the logging profile to remove.</td>
</tr>
</tbody>
</table>

Table 8.19.

8.4.8.2. Response elements

None.

8.4.8.3. Faults

<table>
<thead>
<tr>
<th>Fault type</th>
<th>Cause</th>
</tr>
</thead>
<tbody>
<tr>
<td>tl:LoggingProfileNotFound</td>
<td>The logging profile profileName does not exist on the server</td>
</tr>
</tbody>
</table>

Table 8.20.

8.4.8.4. WSDL Definition

```xml
<!-- operation request element -->
<element name="removeLoggingProfile">
  <complexType>
    <sequence>
      <element name="profileName" type="xs:string" />
    </sequence>
  </complexType>
</element>

<!-- operation response element -->
<element name="removeLoggingProfileResponse">
  <complexType>
    <sequence>
    </sequence>
  </complexType>
</element>
```

8.4.9. saveTransaction

Saves the current transaction state as [BSI TR-03135]-compliant XML data the via a logging provider. If called, MUST be called between beginTransaction and endTransaction.

In order to save log data from an HLDC workflow, it is RECOMMENDED to call saveTransaction only after workflow execution has stopped to ensure that the log data is complete with respect to the workflow.

Workflow execution has stopped if either of the following is true:

- getWorkflowFeedback or getWorkflowFeedbackById return workflowStatus finished or cancelled
- cancelWorkflow was called
- endWorkflow was called.

Otherwise, the server MAY fail the call with InProgress fault.

On success, the transaction MUST be flagged as read-only to prevent further modification.

The server MAY log the data asynchronously and return the call of saveTransaction before the actual logging has finished.
8.4.9.1. Request elements

<table>
<thead>
<tr>
<th>Element name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>transactionID</td>
<td>tl:UUID</td>
</tr>
<tr>
<td></td>
<td>A valid transaction ID.</td>
</tr>
<tr>
<td>provider?</td>
<td>tl:string</td>
</tr>
<tr>
<td></td>
<td>The name of the logging provider which is used for the save operation. If empty, the server MAY use a default logging provider. The names of the logging providers are server-defined. They can be queried if the server provides the optional function getAllLoggingProviders().</td>
</tr>
<tr>
<td>providerParam?</td>
<td>xs:string</td>
</tr>
<tr>
<td></td>
<td>Provider-specific parameter. MAY be empty. The server MAY return an error if the value of providerParam is invalid for provider. The server MUST accept empty providerParam for the default logging provider. The server MAY ignore providerParam if provider is empty.</td>
</tr>
<tr>
<td>errorCode</td>
<td>xs:int</td>
</tr>
<tr>
<td></td>
<td>Application-defined error code which is copied to the [BSI TR-03135]-XML. If there was no error, the errorCode MUST be set to 0.</td>
</tr>
<tr>
<td>loggingProfile</td>
<td>xs:string</td>
</tr>
<tr>
<td></td>
<td>Name of the logging profile to use.</td>
</tr>
</tbody>
</table>

Table 8.21.

8.4.9.2. Response elements

None.

8.4.9.3. Faults

<table>
<thead>
<tr>
<th>Fault type</th>
<th>Cause</th>
</tr>
</thead>
<tbody>
<tr>
<td>t1:InProgress</td>
<td>The transaction is currently being written to and cannot be write-protected.</td>
</tr>
<tr>
<td>t1:InvalidLoggingParameter</td>
<td>Value of providerParam is invalid for the selected provider.</td>
</tr>
<tr>
<td>t1:InvalidLoggingProvider</td>
<td>Value of provider is invalid.</td>
</tr>
<tr>
<td>t1:InvalidTransactionId</td>
<td>Value of transactionID is invalid or has expired.</td>
</tr>
<tr>
<td>t1:LoggingFailed</td>
<td>The server was unable to store the data. This fault MAY be omitted if the server processes the logging call asynchronously.</td>
</tr>
<tr>
<td>t1:LoggingProfileNotFound</td>
<td>The requested logging profile loggingProfile does not exist on the server.</td>
</tr>
</tbody>
</table>

Table 8.22.

8.4.9.4. WSDL Definition

```xml
<complexType>
  <sequence>
    <element name="transactionID" type="tl:UUID" />
    <element name="provider" type="xs:string" minOccurs="0" />
    <element name="providerParam" type="xs:string" minOccurs="0" />
    <element name="errorCode" type="xs:int" />
    <element name="loggingProfile" type="xs:string" />
  </sequence>
</complexType>
```
8.4.10. saveTransactionXML

Saves a complete transaction XML document via a logging provider.

The server MAY log the data asynchronously and return the call of saveTransactionXML before the actual logging has finished.

8.4.10.1. Request elements

<table>
<thead>
<tr>
<th>Element name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>transactionXML</td>
<td>A valid XML-formatted log file according to [BSI TR-03135-1].</td>
</tr>
<tr>
<td>containsExtendedData?</td>
<td>t1:boolean</td>
</tr>
<tr>
<td></td>
<td>Signifies if the XML document contains extended data according to version 1.x of [BSI TR-03135], if the XML complies to version 1.x of [BSI TR-03135].</td>
</tr>
<tr>
<td>provider?</td>
<td>xs:string</td>
</tr>
<tr>
<td></td>
<td>The name of the logging provider which is used for the save operation. If empty, the server MAY use a default logging provider. The names of the logging providers are server-defined. They can be queried if the server provides the optional function getAllLoggingProviders.</td>
</tr>
<tr>
<td>providerParam?</td>
<td>xs:string</td>
</tr>
<tr>
<td></td>
<td>Provider-specific parameter. MAY be empty. The server MAY return an error if the value of providerParam is invalid for provider. The server MUST accept empty providerParam for the default logging provider. The server MAY ignore providerParam if provider is empty.</td>
</tr>
</tbody>
</table>

8.4.10.2. Response elements

None.

8.4.10.3. Faults

<table>
<thead>
<tr>
<th>Fault type</th>
<th>Cause</th>
</tr>
</thead>
<tbody>
<tr>
<td>t1:InvalidLoggingParameter</td>
<td>Value of providerParam is invalid for the selected provider.</td>
</tr>
<tr>
<td>t1:InvalidLoggingProvider</td>
<td>Value of provider is invalid.</td>
</tr>
<tr>
<td>t1:LoggingFailed</td>
<td>The server was unable to store the data. This fault MAY be omitted if the server processes the logging call asynchronously.</td>
</tr>
</tbody>
</table>

8.4.10.4. WSDL Definition

```xml
<element name="saveTransactionXML">
    <complexType>
        <sequence>
            ...
        </sequence>
    </complexType>
</element>
```
8.4.11. setSystemInformation
Saves a complete transaction XML document via a logging provider.

8.4.11.1. Request elements

<table>
<thead>
<tr>
<th>Element name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>transactionID</td>
<td>tl:UUID</td>
</tr>
<tr>
<td>vendor</td>
<td>tl:boolean</td>
</tr>
<tr>
<td>name</td>
<td>xs:string</td>
</tr>
<tr>
<td>version</td>
<td>xs:string</td>
</tr>
</tbody>
</table>

Table 8.25.

8.4.11.2. Response elements
None.

8.4.11.3. Faults

<table>
<thead>
<tr>
<th>Fault type</th>
<th>Cause</th>
</tr>
</thead>
<tbody>
<tr>
<td>tl:InvalidTransactionId</td>
<td>Value of transactionID is invalid or has expired.</td>
</tr>
<tr>
<td>tl:ReadOnly</td>
<td>The transaction is write-protected and must not be modified.</td>
</tr>
</tbody>
</table>

Table 8.26.

8.4.11.4. WSDL Definition

```xml
<element name="setSystemInformation">
  <complexType>
    <sequence>
      <element name="transactionID" type="tl:UUID" />
      <element name="vendor" type="xs:string" />
      <element name="name" type="xs:string" />
      <element name="version" type="xs:string" />
    </sequence>
  </complexType>
</element>
```
<element name="setSystemInformationResponse">
  <complexType>
    <sequence/>
  </complexType>
</element>
# List of Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>BAC</td>
<td>Basic Access Control</td>
</tr>
<tr>
<td>CA</td>
<td>Chip Authentication</td>
</tr>
<tr>
<td>CAN</td>
<td>Card Access Number</td>
</tr>
<tr>
<td>eID</td>
<td>Electronic Identity Document</td>
</tr>
<tr>
<td>HLDC</td>
<td>High Level Document Check</td>
</tr>
<tr>
<td>ICAO</td>
<td>International Civil Aviation Organization</td>
</tr>
<tr>
<td>MRTD</td>
<td>Machine Readable Travel Document</td>
</tr>
<tr>
<td>MRZ</td>
<td>Machine Readable Zone</td>
</tr>
<tr>
<td>PACE</td>
<td>Password Authenticated Connection Establishment</td>
</tr>
<tr>
<td>PNG</td>
<td>Portable Network Graphics</td>
</tr>
<tr>
<td>RFID</td>
<td>radio-frequency identification</td>
</tr>
<tr>
<td>SOAP</td>
<td>Simple Object Access Protocol</td>
</tr>
<tr>
<td>TA</td>
<td>Terminal Authentication</td>
</tr>
<tr>
<td>WSDL</td>
<td>Web Services Description Language</td>
</tr>
<tr>
<td>XML</td>
<td>Extensible Markup Language</td>
</tr>
<tr>
<td>XSD</td>
<td>XML Schema Definition</td>
</tr>
<tr>
<td>XSLT</td>
<td>Extensible Stylesheet Language Transformation</td>
</tr>
</tbody>
</table>
Bibliography


[RFC2119] Request For Comments Editor (RFC), Bradner, Scott: Key words for use in RFCs to indicate requirement levels, RFC 2119, 1997.