

TxToolbox for ODX Manual

Version: 1.5
Issued: 2006-06-09
State: Release

Revision History

Version	Date	Editor	Revisions / Comment
1.0	2005-11-25	Alexander Heinrich	Initial version
1.1	2005-11-30	Alexander Heinrich	Batch mode and typo
1.2	2005-12-02	Alexander Heinrich	Cover redesigned and disclaimer added
1.3	2005-12-14	Alexander Heinrich	Chapter 8 – Java/C++ API
1.4	2005-12-21	Alexander Heinrich	Readapted for changed GUI, TxDiff rules introduced, Batch API: Correlation of -odx and -pdx parameters
1.5	2006-06-09	Alexander Heffner	Description of Java/C++ API adapted

Disclaimer

All information provided in this document comes without any warranties, expressed or implied, of quality, correctness, performance, merchantability, fitness for a particular purpose or non-infringement.

1 Table of Contents

TxToolbox for ODX Manual	i
Revision History	1
Disclaimer	1
1 Table of Contents	2
2 The Application	4
2.1 System Requirements	4
2.2 TxCheck	5
2.2.1 TxCheck Rules	5
2.3 TxDiff	5
2.3.1 TxDiff Rules	5
2.4 Projects	6
3 The Menu Bar	7
3.1 The File Menu	7
3.1.1 Open ODX Files	7
3.1.2 Open PDX File	7
3.1.3 Open Project	7
3.1.4 Save Project	8
3.1.5 Close	8
3.1.6 Exit	8
3.2 The Edit Menu	8
3.2.1 Set Active File	8
3.3 The Checker Menu	9
3.3.1 Check File	9
3.3.2 Check All Files	9
3.3.3 Report CheckResult	9
3.3.4 Report RuleSet	9
3.3.5 Reload Rules	10
3.4 The Differ Menu	10
3.4.1 Compare	10
3.4.2 Report Result	11
3.5 The Settings Menu	11
3.5.1 Preferences	11
3.6 The View Menu	12
3.6.1 Checker	12
3.6.2 Differ	12

3.7	The Help Menu	12
3.7.1	Help	12
3.7.2	Info	12
4	The Toolbar	13
5	The Checker View	14
5.1	File Panel	14
5.2	Console	15
5.3	Results Panel.....	15
5.3.1	Messages Tab	15
5.3.2	Rule Set Tab	16
6	The Differ View	17
6.1	Project Panels.....	17
6.2	Results Panel.....	18
6.2.1	Messages Tab	18
6.2.2	DIAG-SERVICES / DATA-OBJECTS.....	18
6.3	Details Panel	18
6.3.1	Console	18
6.3.2	Tree Views	18
6.3.3	Two-Tree View	19
6.3.4	Single-Tree View.....	20
7	The Formatter	21
8	Batch Mode	22
8.1	Syntax	22
8.2	Examples.....	23
9	The Java/C++ API	24
9.1	The Checker API	24
9.2	Creating Projects.....	25
10	Licenses	27
11	References	28

2 The Application

The TxToolbox application is a Java™-based toolset for developers of ASAM MCD-2D(ODX) / ISO22901 documents and provides the user with functionality for verifying the compliance of such documents to the ODX XML schema and a set of external (pluggable) rules (→ [TxCheck](#)) as well as for comparing documents with each other (→ [TxDiff](#)).

The TxCheck tool comes with a set of rules that were defined by the ODX Joint Expert Group (JEG) and that are part of the ODX standard. These rules can be easily recognized by the prefix ASAM in the filename - their numbering is analogous to that of the rule definitions in the standard.

Developers can extend or tailor the way the TxDiff tool compares documents by providing their own rules for TxDiff (Note that these are different rules than for TxCheck!).

2.1 System Requirements

TxToolbox should be run on a system with at least 512 MB RAM and a 1.6 GHz CPU. While the application may also run on less powerful systems, performance may suffer.

Since the application is based upon Java™ technology, it should be able to run on any system that has a Java Runtime Environment (JRE) installed. However, the TxToolbox package includes its own JREs for MS Windows™ and UNIX platforms: these are installed alongside any existing JREs.

2.2 TxCheck

Use: Verify ODX documents.

This tool can be selected from the View menu in the menu bar.

2.2.1 TxCheck Rules

TxCheck's verification algorithm is based upon external rules. Rules are independent Java™ classes which TxCheck loads from a specific directory (→ [Preferences...](#)). When a check is performed on an ODX document, the tool passes the document to each of the rules one after another; the rules on their parts analyze the passed document and perform the actual compliance checking. If a rule detects a violation, it reports this violation to the TxCheck tool, which notifies the user in form of a message (→ [Messages Tab](#)). A rule usually regards only a very small aspect of a document, e.g., whether the content of a certain element matches a predefined regular expression, whether links reference valid targets or even simple range checking for values. Nevertheless a rule can detect more than one (qualitative and quantitative) violation, meaning that it does not abort document analysis after a violation has been detected (=quantitative) and that it can distinguish between several characteristics of the same violation (=qualitative) which are identified by a sub-code.

2.3 TxDiff

Use: Compare documents or parts of documents (e.g., services or DOPs) to each other and list the differences. TxDiff is able to distinguish between real changes on an object basis and changes that only arise from an altered XML structure but mean no change (e.g., changed IDs). It also presents the compared data in special views that reflect the data structure and highlight changes for quick recognition.

2.3.1 TxDiff Rules

TxDiff's comparison algorithm can be extended and tailored by external rules. Such rules are (cf. TxCheck rules) Java™ based and are dynamically loaded at the time the application starts. They are designed to match a certain group of elements and, if TxDiff encounters elements of that group during a comparison, it will pass them to the appropriate rule instead of handling them itself. TxDiff recognizes rules for XML comparison (→ [XML comparison](#)) and object comparison (→ [Object comparison](#)).

2.4 Projects

A project is basically a set of ODX documents that are identified by their absolute path on the file system. You can save a project (**.cpr** extension) and reload it later. This makes it easier to handle groups of files.

Note: TxDiff makes use of two projects! So if you save a project (➔ Save Project) while in TxDiff view, the operation will save only the currently active one!

3 The Menu Bar

The menu bar is located at the very top of the application window. The entire functionality of the TxToolbox is accessible over the menu bar.



Fig. 3.1 - The Menu Bar

3.1 The File Menu

3.1.1 Open ODX Files...

Use: Invoke a file chooser dialog for selection of one or more ODX documents. These documents are loaded into the project. If the documents selected are numerous or large, the loading may take a few seconds. You can add other documents to the project by repeating this procedure, but it is not possible to open a PDX after other documents have already been loaded. If you try to load a PDX after other documents have been loaded, these documents will be unloaded before the PDX is opened.

A speed button for this menu item is available in the toolbar.. The shortcut for this menu item is Ctrl-O.

3.1.2 Open PDX File

Use: Open a PDX (packed ODX) selected from the file chooser dialog. A PDX may contain multiple ODX documents. The PDX is unpacked and all documents contained are added to the project. You may not have more than one PDX file opened at a time: if the project already contains documents, these will be removed before the PDX is opened. However, it is possible to add other ODX documents to a project when having a PDX opened: execute **Add ODX** or click the corresponding speed button in the toolbar.

A speed button for this menu item is available in the toolbar. The shortcut for this menu item is Ctrl-P.

3.1.3 Open Project

Use: Invoke a file chooser dialog for selection of a project file. The files referenced by this project are loaded into the TxToolbox. Since the files of a project are not stored in its .cpr file, the application might not be able to find the referenced documents (e.g., if they have been deleted or if

the project is being opened from another system where the referenced files do not exist). In this case, an error message is sent to the console for every file missing.

3.1.4 Save Project

Use: Choose a name and destination for a project file (.cpr extension) from the file chooser dialog. This stores the absolute paths to all files that are currently loaded (Important: This does not save the files themselves!). Project files are intended to make it easier to repeatedly handle multiple documents that are possibly located in different directories of the file system.

Note that a project file alone is insufficient - the referenced PDX/ODX documents are always required!

3.1.5 Close

This will unload all currently opened documents. Because documents may be dependent from each other it is not possible to close only a selection of files. Close also clears the console (→ [Console](#)) as well as the results of a previous check or diff.

A speed button for this menu item is available in the toolbar. The shortcut for this menu item is Ctrl-Q.

3.1.6 Exit

Use: Unload all documents, discard all unreported results, and close the TxToolbox application.

The shortcut for this menu item is ESC.

3.2 The Edit Menu

3.2.1 Set Active File

Use: Set the active file of the project when in the Checker view. The active file is the only one checked when the rule checking mechanism is invoked (→ [Check Rules](#)).

It may be easier to select the active file by a simple click on it in the File panel (→ [File Panel](#)).

Note that selecting the active file does not clear the check results. To avoid misinterpreting the results, take a look at the console - it contains detailed information about the last checked file.

The shortcut for this menu item is F9.

3.3 The Checker Menu

This menu is only enabled in the Checker view, which can be selected from the View menu. Otherwise it is disabled (grey).

3.3.1 Check File

Use: Perform a check on the currently active (selected) document. If no document is active, a message to this effect is sent to the console. Only those rules that have been loaded and enabled in the Rule Set tab are checked. If the checked document is large in size or the number of rules is large (i.e., if the checking process takes some time), a progress bar will pop up, showing the status of the rule check. When the check completes, a message is sent to the console, stating whether any errors or warnings have been found and whether (runtime) errors occurred during the checking process. After a successful check, the detailed results (if any) are displayed in the Messages tab of the Checker view. If the Rule Set tab is currently displayed in the Checker view, it automatically switches over to the Messages tab.

If the current project is empty, this menu item is disabled.

A speed button for this menu item is available in the toolbar. The shortcut for this menu item is F7.

3.3.2 Check All Files

Use: Perform a check on all the currently loaded documents.

The shortcut for this menu item is Shift-F7.

3.3.3 Report CheckResult

Use: Save a report after a check about all the rule violations detected (even if there are none) in XML or HTML format. Such a report contains information about the date and time of the check (the timestamp) and the checked file as well as a detailed list of rule violations with the error code, severity (error or warning), and runtime relevance for each together with its textual description and the XPath identifying the flawed elements location in the ODX document.

If the current project is empty or no rule check has been performed yet, this menu item is disabled.

The shortcut for this menu item is Ctrl-Alt-V.

3.3.4 Report RuleSet

Use: Save the details about all currently loaded rules (including the disabled ones) in either an XML or an HTML file. The verification of ODX documents is based upon rules (→ [Rules](#)). Each rule has a unique code, a severity, runtime relevance, and a (brief and full) description of the analysis it performs on the data. Additionally, rules can be enabled/disabled separately.

The shortcut for this menu item is Ctrl-Alt-R.

3.3.5 Reload Rules

TxToolbox loads all the rules that it finds in the specified rule directory (→ [Preferences...](#)) at start-up. Changes that are made to these rules while the application is running have no effect until the application has been restarted or the rules are explicitly reloaded by executing this menu item.

For developers of rules, it can be very helpful to reload a rule set without having to restart the TxToolbox application.

3.4 The Differ Menu

This menu is only enabled in the Differ view, which can be selected from the View menu.

3.4.1 Compare

Use: Invoke the comparison algorithm of the TxDiff tool. The active file from Project1 is compared to the active file from Project2 and the differences (if any) are listed in the Results panel (→ [Results Panel](#)). What TxDiff regards as a difference and how results are presented depends on the comparison algorithm used. TxDiff distinguishes between two algorithms: XML Diff and Object Diff.

3.4.1.1 XML Diff

Use: Compare documents (Files tab in the Project panel) rather than DiagLayers. This algorithm compares XML elements one-by-one and collects all differences except those that derive from textual differences but have the same interpretation on XML level. For example, if the attributes of an element including their values are identical in both documents but their ordering is different, this will not be considered a difference. Another example where the algorithm does not detect a difference would be an element with no content which can be written either as `<ELEMENT></ELEMENT>` or as `<ELEMENT/>`.

3.4.1.2 Object Diff

Use: Compares only DIAG-SERVICES and DOPs (simple and complex) from the selected layers (DiagLayers tab in the Project panel). Unlike XML Diff, this algorithm compares these using a more runtime-oriented approach. That means it looks not only at the XML structure of the (wlog) service but also at its related request, responses, and DOPs. A changed ID or ID-REF (in odxlink) does not necessarily mean a difference to this algorithm as long as the corresponding element is not changed.

3.4.2 Report Result

Use: Save the (message) results of a comparison to an XML, HTML or XLS (MS Excel) file. A comparison report contains the names of the compared files, the current date, and a list of differences. Each difference has a description and one or two XPath's identifying the location of the differing elements.

3.5 The Settings Menu

3.5.1 Preferences

The Preferences dialog is divided into three tabs, each containing the related settings.

The shortcut for the Preferences dialog is Ctrl-Alt-S.

3.5.1.1 General Tab

Use: Set the language of the graphical user interface (GUI). Changes to this setting do not become effective immediately but when the application is started the next time.

3.5.1.2 Differ Tab

Use: Set paths to external rules for file and object comparison and specify whether external rules are to be activated or not. Changes to this setting become effective immediately.

3.5.1.3 Checker Tab

Use: Set the language for reports and error messages as well as the directory where the rules are located.

3.6 The View Menu

Use: Switch between the TxCheck and the TxDiff tools, which do not share a common client area.

3.6.1 Checker

Use: Switch to the Checker view.

3.6.2 Differ

Use: Switch to the Differ view.

3.7 The Help Menu

3.7.1 Help

Use: A browsable online help.

3.7.2 Info

Use: Info dialog providing information about the product version, installed components, and copyright.

4 The Toolbar

The tool bar is located underneath the menu bar. It provides buttons for the most frequently used functions in TxToolbox.

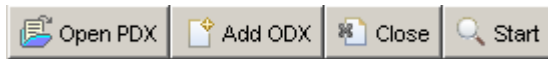



Fig. 4.1 – The Toolbar

The toolbar can be moved to different positions simply by dragging it to one of the four sides of the application window. An empty rectangle is shown at the mouse pointer's position - its frame becomes red if docking is possible at this position; otherwise it remains black. If you drop the toolbar while its frame is black, the toolbar will float. To dock it at its last position, click on the  button.

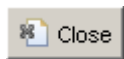
The toolbar contains the following buttons:



Speed button for the Open PDX File... menu item from the File menu



Speed button for the Open ODX Files... menu item from the File menu



Speed button for the Close menu item from the File menu



Speed button for the Check Rules menu item from the Checker menu or for the Compare menu item from the Differ menu, depending on the currently active view: in the Differ view, this button invokes a comparison, in the Checker view a rule check.

5 The Checker View

The Checker view is a set of panels that represent data or invoke functionality related to the rule-based verification of ODX documents.

Switch to the Checker view by choosing the item Checker from the View menu in the menu bar. The client area instantly changes to the view selected.

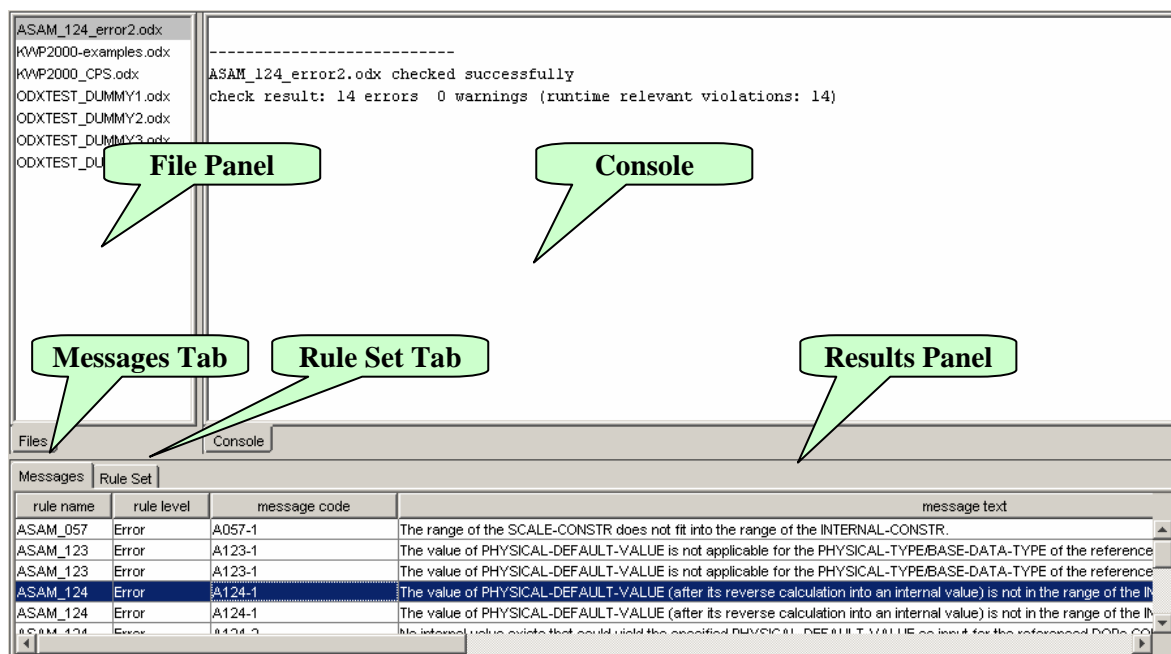


Fig. 5.1 – The Checker view client area

5.1 File Panel

Use: View all currently loaded ODX documents of the open project. The File panel is located at the upper left of the client area. The active file is the highlighted. The File panel is equivalent to Project1 in the Differ view. Thus if you invoke a rule check by clicking the Check Rules speed button while in the Differ view, the document that is checked is the active document from Project1.

5.2 Console

Use: View messages that were generated during runtime, such as error messages, warnings, and results of user actions. The Console is the largest part of the TxCheck client area and is located at the upper right. To clear all messages from the Console, right-click on it and select the option Clear Messages.

5.3 Results Panel

The TxCheck Results panel consists of two tabs: Messages and Rule Set.

5.3.1 Messages Tab

Use: View a list of messages that describe the detected rule violations after a successful check. Each message is described by the name of the rule it was generated by (rule name), its severity (rule level), an error sub-code (message code), its runtime relevance, and the message text. Double-click on a message to bring up a dialog that presents the information in a more legible form. This dialog additionally contains an XPath to the invalid element.

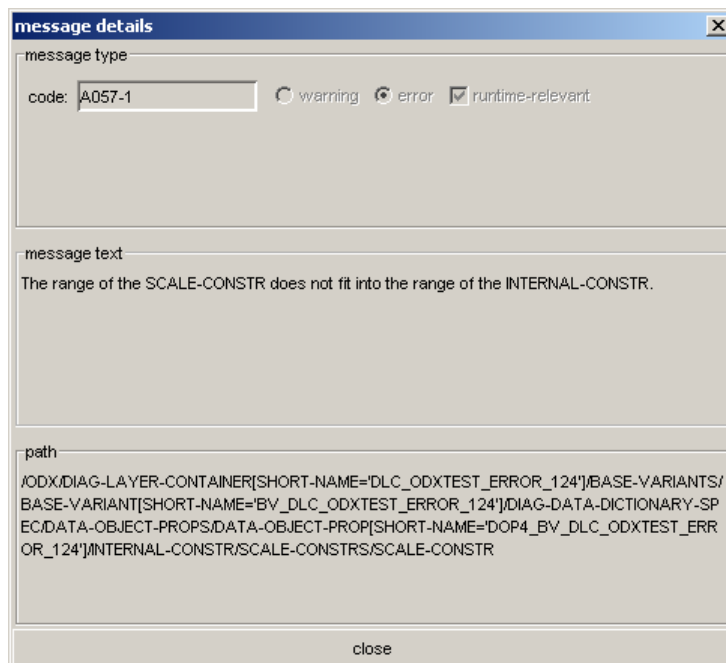


Fig. 5.2 – The message details dialog

5.3.2 Rule Set Tab

Use: View a list of all currently loaded rules. For each rule, a name and a brief description are provided. Rules can be enabled/disabled by checking/unchecking their is active check box. Although disabled rules are not checked during a rule check, they are reported in a rule set report. For more information about a rule, double-click on it: this will bring up a Rule Details dialog containing additional information about its severity and runtime-relevance and a more detailed description.

Messages Rule Set		
rule name	is active	rule description
ASAM_092	<input checked="" type="checkbox"/>	The shortnames of all FLASH-CLASSES referenced via FLASH-CLASS-REF in SESSION-DESC must be unique among each other.
ASAM_093	<input type="checkbox"/>	COMPU-METHODs of DOPs used in a REQUEST must be invertible.
ASAM_096	<input checked="" type="checkbox"/>	PROTOCOL-SNREFS must only occur in DIAG-COMMs that do not reside in a PROTOCOL themselves.
ASAM_097	<input checked="" type="checkbox"/>	For every CASE LOWER- and UPPER-LIMIT must correspond to the SWITCH-KEY.
ASAM_098	<input checked="" type="checkbox"/>	LOWER-LIMIT and UPPER-LIMIT in MUX/CASE must be the same if their datatype is A_UNICODE2STRING.

Fig. 5.3 – The Rule Set Tab

6 The Differ View

The Differ view is a set of panels and buttons that represent data or invoke functionality related to the comparison of data objects.

Data objects can be either complete ODX documents (files) or parts of these (e.g., services or DOPs).

Switch to the Differ view by choosing the item Differ from the View menu in the menu bar. The client area instantly changes to the view selected.

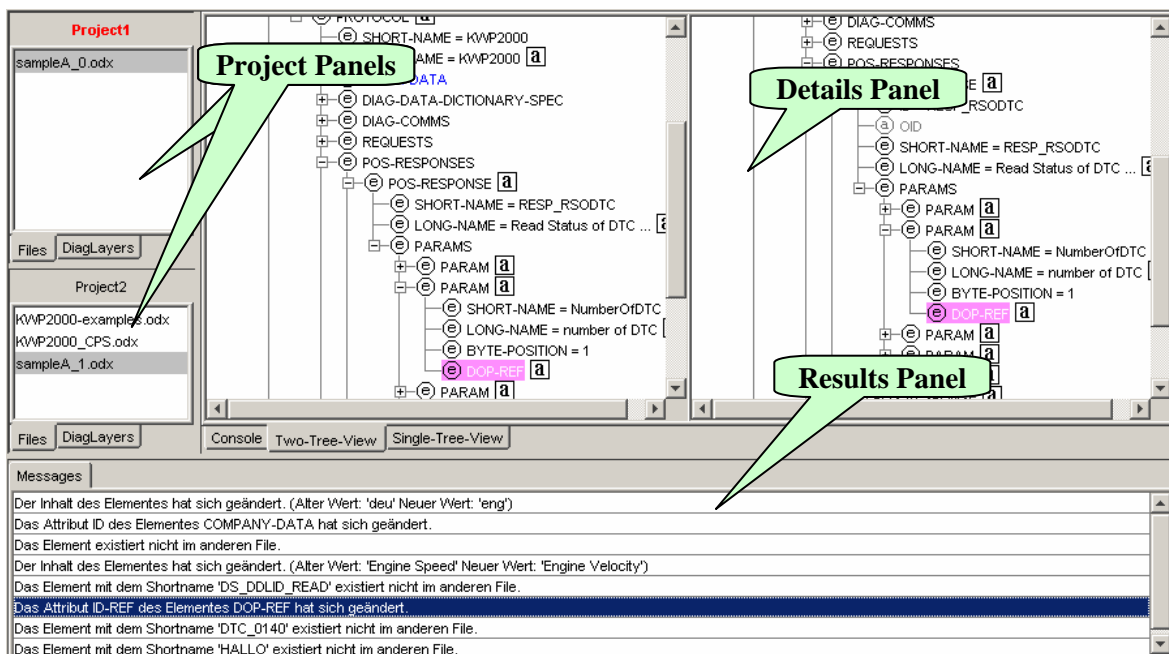


Fig. 6.1 – The Differ view client area

6.1 Project Panels

Use: View all data objects of a project. The Project panels are located on the upper left of the client area: Project1 and Project2. Each project panel has two tabs, Files and DiagLayers, for considering (i.e., comparing) each ODX document as a single and closed instance or for viewing the embedded DiagLayers of all ODX documents in the project, respectively. (→ [Compare](#)). It is not possible to view the files of one project and the DiagLayers of another simultaneously. If you switch to a different tab in one project, the second project will automatically switch to this tab as well.

The heading of the active project (i.e., the one with the focus) is shown in red; the inactive one is shown in black. To make a project the active one, click on it.

Each project has an active data object on which comparisons are performed. - selecting To make a data object in the project panel active, click on it.

To add documents to the active project panel one by one, click **Add ODX**. To remove all documents, click **Close**.

6.2 Results Panel

The Results panel is located at the bottom of the client area. It contains either the Messages tab (if files were compared) or the tabs DIAG-SERVICES and DATA-OBJECTs (if DiagLayers were compared).

6.2.1 Messages Tab

Use: View a list of messages explaining the detected differences of the compared data objects, after a compare operation. If no changes were detected, the Messages tab remains empty. If the Two- or Single-Tree view is active in the Details panel, a double-click on one of the messages jumps to the affected node in the tree.

6.2.2 DIAG-SERVICES / DATA-OBJECTS

If DiagLayers have been compared rather than files, the Result panel contains two tabs, DIAG-SERVICES and DATA-OBJECTS. Use: View information about differences detected during the comparison of services or information about differences detected during the comparison of DOPs, respectively. Each tab contains a list of messages, one for every detected difference. Unlike the Messages tab, messages of these tabs also have a second field containing the name of the service or DOP where the difference was detected.

6.3 Details Panel

The largest (and probably most important) part of the Differ view client area is the Details panel. The Details panel consists of multiple tabs, which are explained in detail below.





6.3.1 Console

The Differ view console corresponds to the TxCheck tool console (→ [Console](#)). This means that console messages generated during a check are also displayed in the Differ console and vice versa.

6.3.2 Tree Views



In addition to the Console tab, one or more (depending on the comparison method) Tree View tabs are available. Tree views are only generated when a comparison has been performed; otherwise

they are empty. They generally visualize the compared data objects as tree structures. Nodes of such trees usually represent an element or an attribute from the ODX documents XML code, but there are exceptions to this rule. The name of a node is (usually) the name of the element or attribute it represents (e.g., the node for an <AUDIENCE> is named AUDIENCE and the node for an attribute OID is also named OID). If an element or attribute has a value (instead of sub-elements), this is appended to the name (e.g., <TROUBLE-CODE>128</TROUBLE-CODE> yields a tree node called TROUBLE-CODE = 666. Sometimes values are too long to be displayed on screen. In these cases only the beginning of the value is shown, followed by.... Move the mouse pointer over the node to pop up a tool tip showing the complete value.

Initially the tree structure of such a view is collapsed. Click the  icon of a node to expand its sub tree, click the  icon to collapse it again. If a node has no  or  icon, it has no sub-elements.

To make it easier to distinguish nodes in a list of homogeneous elements (e.g., PARAMs) without expanding them, short names have been provided in some cases. Move the mouse pointer over such a node to pop up a tool tip with the short name.

The type of a node (element or attribute) can be determined by its icon:

-  Elements
-  Attributes

Differences between data objects are illustrated by colors:

- Black on white Indicates an element that has an identical counter-part (i.e., no difference) in the other data object
- Blue Indicates an element that has no counter-part in the other data object; shown in the tree representing the data object that contains the element (not displayed in the second tree).
- Pink Indicates an element that differs from its existing counter-part


Note: If an element is marked changed (pink) or single (blue), all of its sub-elements share this colour.

The concrete occurrences of Tree views in the Differ view are set out below.

6.3.3 Two-Tree View

A Two-Tree view is a pair of tree structures where the left tree refers to the compared data object from Project1 and the right tree refers to the compared data object of Project2. Two-Tree views exist for both XML Diffs (Files tab selected in project panel) and for Object Diffs (DiagLayers selected in project panel), yet their interests are different.

6.3.3.1 Two-Tree View for XML Comparison

Both trees exactly reflect the XML structure of the compared files. Initially the attributes of all elements are hidden. To unhide (or hide) attributes, click on the  button to the right of each node. If a node has no such button, it means that it has no attributes. Attributes (if shown) are listed in the order they were defined in the schema. Attributes which are defined in the schema but not specified in the document are greyed out. If a default value has been defined in the schema for such an attribute, this is also shown; otherwise the attribute is shown with no value. Attributes that do not occur in the concrete ODX document are never marked changed or inserted.



If a node represents an odxlink and the odxlink refers to an object in the same document, it can be followed by clicking on it while holding **Ctrl** pressed. If the odxlink is valid and refers to an object inside the document, then the node that corresponds to the referenced element is selected in the view.

6.3.3.2 Two-Tree View for Object Comparison



Both trees show the structure of the service or DOP that the user selected in the Result panel. The displayed structure embeds all of its related structures (e.g., requests, responses, DOPs). In this view, attributes are shown unconditionally. Unset attributes are ignored.

6.3.4 Single-Tree View

This view is only available for XML Diffs. It assumes that the compared documents have major similarities and thus displays them as one document, merging their differences together. Elements and attributes that appear white in a Single-Tree view occur in both documents and are identical. Only differing elements are colored (→ [Tree-View](#)). Icons indicate a changed element's origin:

-  Element from Project1
-  Element from Project2

The same applies for single elements (elements that appear in only one document):

-  Element from Project1
-  Element from Project2

7 The Formatter

The formatter is used to generate PDF documents that present information about services, DOPs and other objects contained in ODX documents in a readable format. It is available as a version embedded in the TxToolbox or as a stand-alone application. For details on the features available, see the TxForm manual.

To invoke the Formatter if it is embedded in TxToolbox, switch to the Checker view and click the button **Create PDF**. Note that this button is only active if ODX data has been loaded.

8 Batch Mode

To access the functionality of the TxCheck tool, it is not necessary to start the graphical user interface (GUI). You can check documents directly from a command prompt.

Note: Checking from a command prompt usually yields a *log4j* warning, which can be neglected.

8.1 Syntax

```
java [-Xms<size> -Xmx<size>] -cp .;Checker.jar com.tsystems.checker.Checker
[-pdx <pdx>] -odx <odx>[,<odx>]* -result <out> [-de|-en] [-xml|-html]
```

<pdx>	Path (absolute or relative) to a PDX file
<odx>	Path (absolute or relative) to an ODX file or a filename referring a file inside a PDX
<out>	Path (absolute or relative) of the report file

The above syntax implies that you must specify one or more input files, which can be either one PDX using the `-pdx` parameter or a number of ODX files using the `-odx` parameter and separating files by commas. If you specify the `-pdx` parameter, you must also specify the `-odx` parameter to tell TxCheck which document within the PDX to check. In this case, you only need to pass one (unqualified) filename for `-odx`. If you do not specify the `-pdx` parameter but more than one file for the `-odx` parameter, then the passed filenames must be (either absolutely or relatively) qualified and the Checker will assume the first file as the active one. You must further specify an output file using the `-result` parameter where the check report is to be saved. If the file specified already exists it will be overwritten. The report format can be chosen using the optional parameters `-xml` (XML) or `-html` (HTML) with default XML. The language of the report can be chosen using the optional parameters `-de` (German) or `-en` (English) with default English.

If the checked files are very large, TxCheck may abnormally terminates with an out-of-memory error. In this case, you can assign more memory to the virtual machine using `-Xms` (initial heap size) and `-Xmx` (maximum heap size). Suitable values for these parameters depend on the system the TxCheck tool is being run on. For the system described in [section 2.1](#), meaningful values would be `-Xms100m -Xmx500m`. Generally you should set these values neither too high nor too low. Assigning more memory than physically available will make the virtual machine swap data to the hard disk and performance will decrease significantly. Assigning too little memory will, of course, not solve the out-of-memory problem.

8.2 Examples

1. Check the ODX document **KWP2000_examples.odx** and store the report in **result.xml** using the default settings for output format and language (XML and English).

```
java -cp .;Checker.jar com.tsystems.checker.Checker -odx KWP2000_examples.odx  
-result result.xml
```

2. Check the ODX documents **KWP2000_examples.odx** and **KWP2000_CPS.odx** and save the report in the file **result.html** with output format and language explicitly set to HTML and German.

```
java -cp .;Checker.jar com.tsystems.checker.Checker  
-odx KWP2000_examples.odx,KWP2000_CPS.odx -result result.html -html -de
```

3. Check the ODX documents **KWP2000_examples.odx** contained in the PDX **KWP2000.pdx**.

```
java -cp .;Checker.jar com.tsystems.checker.Checker -pdx KWP2000.pdx -odx  
KWP2000_examples.odx -result result.xml
```

Check the large file **big.odx**.

```
java -Xms100m -Xmx500m -cp .;Checker.jar com.tsystems.checker.Checker -odx big.odx  
-result result.xml
```


9 The Java/C++ API

You can access the complete functionality of the TxCheck tool from your own Java or C++ applications. Although TxCheck is implemented in Java, there is a way to embed a Java Virtual Machine into a C++ application and instantiate a Checker. This is basically managed by the Java Native Interface (JNI).

9.1 The Checker API

The Java class to instantiate is Checker4J. For its interface, refer to the JavaDocs provided with TxToolbox. Instantiation in C++ is managed by proxies. All you need to do is to instantiate a proxy which corresponds to the Java class to be used and call its functions just like in Java. An example is given below, using two code fragments, one written in C++ and one written in Java:

C++	Java
<pre> StaticVmLoader loader(JNI_VERSION_1_2); OptionList list; list.push_back(jace::ClassPath("./lib/txtoolbox.jar")); list.push_back(jace::CustomOption("-Xcheck:jni")); list.push_back(jace::CustomOption("-Xmx128M")); jace::helper::createVm(loader, list, false); IF_Checker4J checker = IF_Checker4J_Factory::newInstance(); typedef JArray<File> FileArray; FileArray files(6); files[0] = File("ASAM_001_error.odx"); files[1] = File("ASAM_001_pass.odx"); files[2] = File("ASAM_003_error.odx"); checker.setRulePath("./rules/"); checker.loadODXFiles(files); checker.setReportFormat("XML"); checker.reportRuleSet(File("ruleset.xml")); typedef JArray<Message> MessageArray; for (int i=0; i < files.length(); i++) { String active = files[i].getAbsolutePath(); checker.setActiveFile(active); </pre>	<pre> Checker4J checker = IF_Checker4J.Factory.newInstance(); File[] files = File[6]; files[0] = File("ASAM_001_error.odx"); files[1] = File("ASAM_001_pass.odx"); files[2] = File("ASAM_003_error.odx"); checker.setRulePath("./rules/"); checker.loadODXFiles(files); checker.setReportFormat("XML"); checker.reportRuleSet(File("ruleset.xml")); for (int i=0; i < 6; i++) { String active = files[i].getName(); checker.setActiveFile(active); checker.check(); File result = </pre>

<pre> checker.check(); File result = File("result"+String::valueOf(JInt(i))+".xml"); checker.reportResult(result); } </pre>	<pre> File("result" + i + ".xml"); checker.reportResult(result); } </pre>
---	---

9.2 Creating Projects

Several steps are required to integrate the C++ API for ASAM ODX Checker into a C++ project.

Most importantly a Java Development Kit (JDK) must be installed on the system - a mere Runtime Environment will not suffice, because headers that are only contained in the JDK but not in the JRE have to be included and libraries imported. Also, the library **jace.lib** or **jaced.lib** (for debugging) from the **lib** directory is required and (of course) the proxy classes that map the Java classes to their native representation. The latter are located in the **dev/c_api/proxies** directory of this package.

Add the **jace** library to the project as well as the **lib/jvm.lib** from the JDK installation directory. The JDK **include** directory must also be on the search path for header files and also its subdirectory in line with the operating system used (e.g., win32). The **proxies/include** directory must be put to this search path as well. Run-Time-Type-Information (RTTI) should be enabled in the compiler settings.

The **proxies/source** directory contains all the native classes that are required for using the Checker. Import the source files of all classes in the workspace that you think you might need. Now you can start utilizing the TxCheck tool in your native application.

Helpful Hints:

- The Javadocs can be consulted for more information about the TxCheck native interface. Apart from minor syntactical differences, the native classes can be used in almost exactly the same way as the original Java classes can. A small exception is that Java fields are mapped to methods in C++, i.e., if the Java class has a public variable **int x** this variable can be accessed via the method **x()** in the C++ class. Primitive and even string return values from a Java method invocation are automatically converted to the equivalent C++ types. There is no need for extra operations.
- It is important to always include all required header files. For example, if a list of Message objects is being retrieved from the TxCheck tool (using **getRuleViolations()**) and these are to be analysed, then the appropriate headers must be included:

```
#include "jace/proxy/com/tsystems/checker/Message.h"

using jace::proxy::com::tsystems::checker::Message;
```


10 Licenses

The TxToolbox Application includes software developed by the following organizations:

The Apache Software Foundation (<http://www.apache.org>)

The DOM4J Project (<http://www.dom4j.org>)

For details about license agreements, visit these sites.

11 References

1. ASAM ODX Schema Version 2.0.1
<http://www.asam.net/xml/odx/2.0.1/odx.xsd>
<http://www.asam.net/xml/odx/2.0.1/odx-xhtml.xsd>
<http://www.asam.net/xml/odx/2.0.1/odx-cc.xsd>
2. ODX Specification ASAM MCD-2D (ODX) 2.0.1
http://www.asam.net/O3_standards_06.php