ABBYY Recognition Server 4 Open API
Open API Overview

The Open API is a component of ABBYY Recognition Server. It can be installed onto any computer in the network. ABBYY Recognition Server provides two types of Open API for integration with external systems: a COM-based API and a Web Services API.

The COM-based API uses RPC technology, and can be used for integration with other systems in the local area network. In case of the COM-based API you need to install required components of ABBYY Recognition Server on the computer where the client application is installed. See Distribution of Applications Using the ABBYY Recognition Server Com-based API Library for details.

The Web service API enables communication with remote systems via HTTP, and therefore allows for remote integration over the Internet. In case of the Web services API you do not need to install any components on the client computer.

To generate WSDL description, connect to http://<ServerName>/Recognition4WS/RSSoapService.asmx?WSDL

Note:

- The Recognition Server Web Service may require very complex configuration, if it is installed on the same computer with Microsoft Office SharePoint Server. It is not recommended to install them on the same computer.
- The Web Service will not work if ASP.NET is not installed on your computer. When installing Microsoft .Net Framework ASP.NET is copied on the computer but not installed. You should install it manually by running:
  C:\WINDOWS\Microsoft.NET\Framework\vX.X.XXX\aspnet_regiis.exe -i (specify the appropriate version instead of vX.X.XXX).

This section will provide detailed descriptions of the Open API objects, their properties, and methods. These objects allow you to create XML Tickets, manage jobs, analyze XML Results, and specify preprocessing, recognition and export parameters.
ABBYY Recognition Server 4 Open API
Recognition Languages

Below is the list of internal names of the recognition languages that are supported in ABBYY Recognition Server 4. Availability of a recognition language depends on whether you have the corresponding modules installed. ABBYY Recognition Server provides its own system dictionaries for the languages that have built-in dictionary support.

The **Languages** property of the **RecognitionParams** object specifies a collection of recognition languages. Elements of this collection must be internal names from the first column.

<table>
<thead>
<tr>
<th>Internal name</th>
<th>Recognition language</th>
<th>Full dictionary support available</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abkhaz</td>
<td>Abkhaz</td>
<td></td>
</tr>
<tr>
<td>Adyghe</td>
<td>Adyghe</td>
<td></td>
</tr>
<tr>
<td>Afrikaans</td>
<td>Afrikaans</td>
<td></td>
</tr>
<tr>
<td>Agul</td>
<td>Agul</td>
<td></td>
</tr>
<tr>
<td>Albanian</td>
<td>Albanian</td>
<td></td>
</tr>
<tr>
<td>Altaic</td>
<td>Altaic</td>
<td></td>
</tr>
<tr>
<td>Arabic</td>
<td>Arabic</td>
<td></td>
</tr>
<tr>
<td>ArmenianEastern</td>
<td>Armenian (Eastern)</td>
<td>+</td>
</tr>
<tr>
<td>ArmenianGrabar</td>
<td>Armenian (Grabar)</td>
<td>+</td>
</tr>
<tr>
<td>ArmenianWestern</td>
<td>Armenian (Western)</td>
<td>+</td>
</tr>
<tr>
<td>Awar</td>
<td>Avar</td>
<td></td>
</tr>
<tr>
<td>Language</td>
<td>Text</td>
<td></td>
</tr>
<tr>
<td>------------------</td>
<td>----------------</td>
<td></td>
</tr>
<tr>
<td>Aymara</td>
<td>Aymara</td>
<td></td>
</tr>
<tr>
<td>AzeriCyrillic</td>
<td>Azerbaijani (Cyrillic)</td>
<td></td>
</tr>
<tr>
<td>AzeriLatin</td>
<td>Azerbaijani (Latin)</td>
<td></td>
</tr>
<tr>
<td>Bashkir</td>
<td>Bashkir</td>
<td></td>
</tr>
<tr>
<td>Basque</td>
<td>Basque</td>
<td></td>
</tr>
<tr>
<td>Belarusian</td>
<td>Belarussian</td>
<td></td>
</tr>
<tr>
<td>Bemba</td>
<td>Bemba</td>
<td></td>
</tr>
<tr>
<td>Blackfoot</td>
<td>Blackfoot</td>
<td></td>
</tr>
<tr>
<td>Breton</td>
<td>Breton</td>
<td></td>
</tr>
<tr>
<td>Bugotu</td>
<td>Bugotu</td>
<td></td>
</tr>
<tr>
<td>Bulgarian</td>
<td>Bulgarian</td>
<td></td>
</tr>
<tr>
<td>Buryat</td>
<td>Buryat</td>
<td></td>
</tr>
<tr>
<td>Catalan</td>
<td>Catalan</td>
<td></td>
</tr>
<tr>
<td>Chamorro</td>
<td>Chamorro</td>
<td></td>
</tr>
<tr>
<td>Chechen</td>
<td>Chechen</td>
<td></td>
</tr>
<tr>
<td>ChinesePRC*</td>
<td>Chinese Simplified</td>
<td></td>
</tr>
<tr>
<td>Language</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>--------------------------------</td>
<td>------------------------------</td>
<td></td>
</tr>
<tr>
<td>Chinese Taiwan*</td>
<td>Chinese Traditional</td>
<td></td>
</tr>
<tr>
<td>Chukcha</td>
<td>Chukchee</td>
<td></td>
</tr>
<tr>
<td>Chuvash</td>
<td>Chuvash</td>
<td></td>
</tr>
<tr>
<td>Corsican</td>
<td>Corsican</td>
<td></td>
</tr>
<tr>
<td>Crimean Tatar</td>
<td>Crimean Tatar</td>
<td></td>
</tr>
<tr>
<td>Croatian</td>
<td>Croatian</td>
<td></td>
</tr>
<tr>
<td>Crow</td>
<td>Crow</td>
<td></td>
</tr>
<tr>
<td>Czech</td>
<td>Czech</td>
<td></td>
</tr>
<tr>
<td>Danish</td>
<td>Danish</td>
<td></td>
</tr>
<tr>
<td>Dargwa</td>
<td>Dargwa</td>
<td></td>
</tr>
<tr>
<td>Dungan</td>
<td>Dungan</td>
<td></td>
</tr>
<tr>
<td>Dutch</td>
<td>Dutch (Netherlands)</td>
<td></td>
</tr>
<tr>
<td>Dutch Belgian</td>
<td>Dutch (Belgium)</td>
<td></td>
</tr>
<tr>
<td>English</td>
<td>English</td>
<td></td>
</tr>
<tr>
<td>Eskimo Cyrillic</td>
<td>Eskimo (Cyrillic)</td>
<td></td>
</tr>
<tr>
<td>Language</td>
<td>Language</td>
<td></td>
</tr>
<tr>
<td>----------------</td>
<td>----------------</td>
<td></td>
</tr>
<tr>
<td>Eskimo Latin</td>
<td>Eskimo (Latin)</td>
<td></td>
</tr>
<tr>
<td>Esperanto</td>
<td>Esperanto</td>
<td></td>
</tr>
<tr>
<td>Estonian</td>
<td>Estonian</td>
<td></td>
</tr>
<tr>
<td>Even</td>
<td>Even</td>
<td></td>
</tr>
<tr>
<td>Evenki</td>
<td>Evenki</td>
<td></td>
</tr>
<tr>
<td>Faeroese</td>
<td>Faeroese</td>
<td></td>
</tr>
<tr>
<td>Fijian</td>
<td>Fijian</td>
<td></td>
</tr>
<tr>
<td>Finnish</td>
<td>Finnish</td>
<td></td>
</tr>
<tr>
<td>French</td>
<td>French</td>
<td></td>
</tr>
<tr>
<td>Frisian</td>
<td>Frisian</td>
<td></td>
</tr>
<tr>
<td>Friulian</td>
<td>Friulian</td>
<td></td>
</tr>
<tr>
<td>Gaelic Scottish</td>
<td>Scottish Gaelic</td>
<td></td>
</tr>
<tr>
<td>Gagauz</td>
<td>Gagauz</td>
<td></td>
</tr>
<tr>
<td>Galician</td>
<td>Galician</td>
<td></td>
</tr>
<tr>
<td>Ganda</td>
<td>Ganda</td>
<td></td>
</tr>
<tr>
<td>German</td>
<td>German</td>
<td></td>
</tr>
<tr>
<td>Language</td>
<td>Code</td>
<td></td>
</tr>
<tr>
<td>------------------------</td>
<td>---------------</td>
<td></td>
</tr>
<tr>
<td>German (new spelling)</td>
<td>GermanNewSpelling</td>
<td></td>
</tr>
<tr>
<td>German (Luxembourg)</td>
<td>GermanLuxembourg</td>
<td></td>
</tr>
<tr>
<td>Greek</td>
<td>Greek</td>
<td></td>
</tr>
<tr>
<td>Guarani</td>
<td>Guarani</td>
<td></td>
</tr>
<tr>
<td>Hani</td>
<td>Hani</td>
<td></td>
</tr>
<tr>
<td>Hausa</td>
<td>Hausa</td>
<td></td>
</tr>
<tr>
<td>Hawaiian</td>
<td>Hawaiian</td>
<td></td>
</tr>
<tr>
<td>Hebrew</td>
<td>Hebrew</td>
<td></td>
</tr>
<tr>
<td>Hungarian</td>
<td>Hungarian</td>
<td></td>
</tr>
<tr>
<td>Icelandic</td>
<td>Icelandic</td>
<td></td>
</tr>
<tr>
<td>Ido</td>
<td>Ido</td>
<td></td>
</tr>
<tr>
<td>Indonesian</td>
<td>Indonesian</td>
<td></td>
</tr>
<tr>
<td>Ingush</td>
<td>Ingush</td>
<td></td>
</tr>
<tr>
<td>Interlingua</td>
<td>Interlingua</td>
<td></td>
</tr>
<tr>
<td>Irish</td>
<td>Irish</td>
<td></td>
</tr>
<tr>
<td>Language</td>
<td>Language</td>
<td></td>
</tr>
<tr>
<td>---------------------</td>
<td>-------------------</td>
<td></td>
</tr>
<tr>
<td>Italian</td>
<td>Italian</td>
<td></td>
</tr>
<tr>
<td>Japanese*</td>
<td>Japanese</td>
<td></td>
</tr>
<tr>
<td>Kabardian</td>
<td>Kabardian</td>
<td></td>
</tr>
<tr>
<td>Kalmyk</td>
<td>Kalmyk</td>
<td></td>
</tr>
<tr>
<td>KarachayBalkar</td>
<td>Karachay-Balkar</td>
<td></td>
</tr>
<tr>
<td>Karakalpak</td>
<td>Karakalpak</td>
<td></td>
</tr>
<tr>
<td>Kasub</td>
<td>Kasub</td>
<td></td>
</tr>
<tr>
<td>Kawa</td>
<td>Kawa</td>
<td></td>
</tr>
<tr>
<td>Kazakh</td>
<td>Kazakh</td>
<td></td>
</tr>
<tr>
<td>Khakas</td>
<td>Khakas</td>
<td></td>
</tr>
<tr>
<td>Khanty</td>
<td>Khanty</td>
<td></td>
</tr>
<tr>
<td>Kikuyu</td>
<td>Kikuyu</td>
<td></td>
</tr>
<tr>
<td>Kirgiz</td>
<td>Kirghiz</td>
<td></td>
</tr>
<tr>
<td>Kongo</td>
<td>Kongo</td>
<td></td>
</tr>
<tr>
<td>Korean*</td>
<td>Korean</td>
<td></td>
</tr>
<tr>
<td>KoreanHangul*</td>
<td>Korean (Hangul)</td>
<td></td>
</tr>
<tr>
<td>Language</td>
<td>Language</td>
<td></td>
</tr>
<tr>
<td>----------</td>
<td>----------</td>
<td></td>
</tr>
<tr>
<td>Koryak</td>
<td>Koryak</td>
<td></td>
</tr>
<tr>
<td>Kpelle</td>
<td>Kpelle</td>
<td></td>
</tr>
<tr>
<td>Kumyk</td>
<td>Kumyk</td>
<td></td>
</tr>
<tr>
<td>Kurdish</td>
<td>Kurdish</td>
<td></td>
</tr>
<tr>
<td>Lak</td>
<td>Lak</td>
<td></td>
</tr>
<tr>
<td>Lappish</td>
<td>Sami (Lappish)</td>
<td></td>
</tr>
<tr>
<td>Latin</td>
<td>Latin</td>
<td></td>
</tr>
<tr>
<td>Latvian</td>
<td>Latvian</td>
<td></td>
</tr>
<tr>
<td>LatvianGothic</td>
<td>Latvian Gothic</td>
<td></td>
</tr>
<tr>
<td>Lezgin</td>
<td>Lezgi</td>
<td></td>
</tr>
<tr>
<td>Lithuanian</td>
<td>Lithuanian</td>
<td></td>
</tr>
<tr>
<td>Luba</td>
<td>Luba</td>
<td></td>
</tr>
<tr>
<td>Macedonian</td>
<td>Macedonian</td>
<td></td>
</tr>
<tr>
<td>Malagasy</td>
<td>Malagasy</td>
<td></td>
</tr>
<tr>
<td>Malay</td>
<td>Malay (Malaysian)</td>
<td></td>
</tr>
<tr>
<td>Malinke</td>
<td>Malinke</td>
<td></td>
</tr>
</tbody>
</table>
Maltese
Mansi
Maori
Mari
Maya
Miao
Minangkabau
Mohawk
Moldavian
Mongol
Mordvin
Nahuatl
Nenets
Nivkh
Nogay
NorwegianNynorsk + NorwegianBokmal
<table>
<thead>
<tr>
<th>Language</th>
<th>Name</th>
<th>+</th>
</tr>
</thead>
<tbody>
<tr>
<td>NorwegianBokmal</td>
<td>Norwegian (Bokmal)</td>
<td></td>
</tr>
<tr>
<td>NorwegianNynorsk</td>
<td>Norwegian (Nynorsk)</td>
<td></td>
</tr>
<tr>
<td>Nyanja</td>
<td>Nyanja</td>
<td></td>
</tr>
<tr>
<td>Occidental</td>
<td>Occidental</td>
<td></td>
</tr>
<tr>
<td>Ojibway</td>
<td>Ojibway</td>
<td></td>
</tr>
<tr>
<td>OldEnglish</td>
<td>Old English</td>
<td></td>
</tr>
<tr>
<td>OldFrench</td>
<td>Old French</td>
<td></td>
</tr>
<tr>
<td>OldGerman</td>
<td>Old German</td>
<td></td>
</tr>
<tr>
<td>OldItalian</td>
<td>Old Italian</td>
<td></td>
</tr>
<tr>
<td>OldSpanish</td>
<td>Old Spanish</td>
<td></td>
</tr>
<tr>
<td>Ossetic</td>
<td>Ossetian</td>
<td></td>
</tr>
<tr>
<td>Papiamento</td>
<td>Papiamento</td>
<td></td>
</tr>
<tr>
<td>PidginEnglish</td>
<td>Tok Pisin</td>
<td></td>
</tr>
<tr>
<td>Polish</td>
<td>Polish</td>
<td></td>
</tr>
<tr>
<td>PortugueseBrazilian</td>
<td>Portuguese (Brazil)</td>
<td></td>
</tr>
<tr>
<td>Language</td>
<td>Name</td>
<td></td>
</tr>
<tr>
<td>------------------------</td>
<td>-----------------------------</td>
<td></td>
</tr>
<tr>
<td>Portuguese Standard</td>
<td>Portuguese (Portugal)</td>
<td></td>
</tr>
<tr>
<td>Provencal</td>
<td>Occitan</td>
<td></td>
</tr>
<tr>
<td>Quechua</td>
<td>Quechua</td>
<td></td>
</tr>
<tr>
<td>RhaetoRomanic</td>
<td>Rhaeto-Romanic</td>
<td></td>
</tr>
<tr>
<td>Romanian</td>
<td>Romanian</td>
<td></td>
</tr>
<tr>
<td>Romany</td>
<td>Romany</td>
<td></td>
</tr>
<tr>
<td>Ruanda</td>
<td>Rwanda</td>
<td></td>
</tr>
<tr>
<td>Rundi</td>
<td>Rundi</td>
<td></td>
</tr>
<tr>
<td>RussianOldSpelling</td>
<td>Russian (old spelling)</td>
<td></td>
</tr>
<tr>
<td>Russian</td>
<td>Russian</td>
<td></td>
</tr>
<tr>
<td>Samoan</td>
<td>Samoan</td>
<td></td>
</tr>
<tr>
<td>Selkup</td>
<td>Selkup</td>
<td></td>
</tr>
<tr>
<td>SerbianCyrillic</td>
<td>Serbian (Cyrillic)</td>
<td></td>
</tr>
<tr>
<td>SerbianLatin</td>
<td>Serbian (Latin)</td>
<td></td>
</tr>
<tr>
<td>Shona</td>
<td>Shona</td>
<td></td>
</tr>
<tr>
<td>Sioux</td>
<td>Sioux (Dakota)</td>
<td></td>
</tr>
<tr>
<td>Language</td>
<td>Language</td>
<td></td>
</tr>
<tr>
<td>----------</td>
<td>----------</td>
<td></td>
</tr>
<tr>
<td>Slovak</td>
<td>Slovak</td>
<td></td>
</tr>
<tr>
<td>Slovenian</td>
<td>Slovenian</td>
<td></td>
</tr>
<tr>
<td>Somali</td>
<td>Somali</td>
<td></td>
</tr>
<tr>
<td>Sorbian</td>
<td>Sorbian</td>
<td></td>
</tr>
<tr>
<td>Sotho</td>
<td>Sotho</td>
<td></td>
</tr>
<tr>
<td>Spanish</td>
<td>Spanish</td>
<td></td>
</tr>
<tr>
<td>Sunda</td>
<td>Sunda</td>
<td></td>
</tr>
<tr>
<td>Swahili</td>
<td>Swahili</td>
<td></td>
</tr>
<tr>
<td>Swazi</td>
<td>Swazi</td>
<td></td>
</tr>
<tr>
<td>Swedish</td>
<td>Swedish</td>
<td></td>
</tr>
<tr>
<td>Tabassaran</td>
<td>Tabassaran</td>
<td></td>
</tr>
<tr>
<td>Tagalog</td>
<td>Tagalog</td>
<td></td>
</tr>
<tr>
<td>Tahitian</td>
<td>Tahitian</td>
<td></td>
</tr>
<tr>
<td>Tajik</td>
<td>Tajik</td>
<td></td>
</tr>
<tr>
<td>Tatar</td>
<td>Tatar</td>
<td></td>
</tr>
<tr>
<td>Language</td>
<td>Code</td>
<td></td>
</tr>
<tr>
<td>------------------</td>
<td>---------------</td>
<td></td>
</tr>
<tr>
<td>Thai</td>
<td>Thai</td>
<td></td>
</tr>
<tr>
<td>Tinpo</td>
<td>Jingpo</td>
<td></td>
</tr>
<tr>
<td>Tongan</td>
<td>Tongan</td>
<td></td>
</tr>
<tr>
<td>Tswana</td>
<td>Tswana</td>
<td></td>
</tr>
<tr>
<td>Tun</td>
<td>Tun</td>
<td></td>
</tr>
<tr>
<td>Turkish</td>
<td>Turkish</td>
<td></td>
</tr>
<tr>
<td>Turkmen</td>
<td>Turkmen</td>
<td></td>
</tr>
<tr>
<td>Tuvin</td>
<td>Tuvininian</td>
<td></td>
</tr>
<tr>
<td>Udmurt</td>
<td>Udmurt</td>
<td></td>
</tr>
<tr>
<td>Uighur Cyrillic</td>
<td>Uighur (Cyrillic)</td>
<td></td>
</tr>
<tr>
<td>Uighur Latin</td>
<td>Uighur (Latin)</td>
<td></td>
</tr>
<tr>
<td>Ukrainian</td>
<td>Ukrainian</td>
<td></td>
</tr>
<tr>
<td>Uzbek Cyrillic</td>
<td>Uzbek (Cyrillic)</td>
<td></td>
</tr>
<tr>
<td>Uzbek Latin</td>
<td>Uzbek (Latin)</td>
<td></td>
</tr>
<tr>
<td>Vietnamese</td>
<td>Vietnamese</td>
<td></td>
</tr>
<tr>
<td>Visayan</td>
<td>Cebuano</td>
<td></td>
</tr>
<tr>
<td>Language</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>------------</td>
<td>--------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>Welsh</td>
<td>Welsh</td>
<td></td>
</tr>
<tr>
<td>Wolof</td>
<td>Wolof</td>
<td></td>
</tr>
<tr>
<td>Xhosa</td>
<td>Xhosa</td>
<td></td>
</tr>
<tr>
<td>Yakut</td>
<td>Yakut</td>
<td></td>
</tr>
<tr>
<td>Yiddish</td>
<td>Yiddish</td>
<td></td>
</tr>
<tr>
<td>Zapotec</td>
<td>Zapotec</td>
<td></td>
</tr>
<tr>
<td>Zulu</td>
<td>Zulu</td>
<td></td>
</tr>
<tr>
<td>Basic</td>
<td>Basic programming language</td>
<td></td>
</tr>
<tr>
<td>C++</td>
<td>C/C++ programming language</td>
<td></td>
</tr>
<tr>
<td>Cobol</td>
<td>Cobol programming language</td>
<td></td>
</tr>
<tr>
<td>Fortran</td>
<td>Fortran programming language</td>
<td></td>
</tr>
<tr>
<td>Java</td>
<td>Java programming language</td>
<td></td>
</tr>
<tr>
<td>Pascal</td>
<td>Pascal programming language</td>
<td></td>
</tr>
<tr>
<td>Chemistry</td>
<td>Simple chemical formulas</td>
<td></td>
</tr>
<tr>
<td>E13B</td>
<td>For MICR (E-13B) text type</td>
<td></td>
</tr>
</tbody>
</table>
* — When exporting texts in hieroglyphic languages to PDF, ABBYY Recognition Server uses the following fonts:

- PMingLiU and MingLiU for Chinese Traditional,
- SimSun for Chinese Simplified,
- MS Mincho and MS PMincho for Japanese,
- Batang and BatangChe for Korean.

To be able to export texts in East Asian languages to PDF, you must have these fonts installed in Windows OS on all computers which are used as Processing Stations. In Asian versions of Windows these fonts are installed by default. In other versions of Windows you can install them by enabling the "Install files for East Asian languages" option in the Languages tab (Control Panel > Regional and Language Options > Languages).
ABBYY Recognition Server 4 Open API
ABBYY FineReader Internal Format

ABBYY FineReader internal format consists of two parts: image (*.image) and layout (*.layout). An image file in the internal format has three "image planes":

- **Black-and-white** plane. This is a black-and-white copy of the source image.
- **Color** plane. This is a color or gray copy of the source image. If the source image was black-and-white, this page is the same as the "black-and-white" plane.
- **Preview**. A small gray image used for displaying a preview document in the graphical user interface.

A layout file in the internal format contains blocks (image areas of specified type - text, picture, table, barcode) and recognized text.

Files in ABBYY FineReader internal format can be imported into ABBYY FineReader Engine.
ABBYY Recognition Server 4 Open API
Compatibility of ABBYY Recognition Server 4.0 and previous versions

Version 4.0 of the COM API is backward compatible with version 3.5.

Version 4.0 of Web API is also backward compatible with version 3.5, but the address of SOAP service has changed to http://localhost/Recognition4WS/RSSoapService.asmx.

Version 3.5 and 3.0 Release 9 of the API are fully compatible.

Version 3.0 Release 9 and 3.0 Release 8 of the Open API are partially incompatible. The changes are described below.

IXmlResult has a new method JobDocuments(), which returns IJobDocuments – a collection of IJobDocument elements. IJobDocument provides access to a collection of output files belonging to the document and the document properties. Properties of IJobDocument:

- **Name** – document name, formed automatically;
- **OutputDocuments** – a collection of OutputDocument elements, each of which stores the list of files of a certain format and properties of those files;
- **Errors** – a collection of errors that occurred during the document processing;
- **Warnings** - a collection of warnings that occurred during the document processing;
- **Statistics** – information about the number of characters, uncertainly recognized characters and pages in the document;
- **Attributes** – document type, a collection of document attributes and information about whether a manual document indexing was performed;
- **CustomText** – value passed from scripts;
- **BarcodeText** – value of the separation barcode (if the document separation was performed by barcodes).

Old properties left for compatibility, now empty:

- **Attributes** property of the IInputFile object;
- **CustomText** property of the IInputFile object;
- **BarcodeText** property of the IXmlResult object.

**Important!** Starting with the release 9 you have to use new properties under IJobDocuments in order to get the document type and attributes, separation barcode value and custom text value. The old properties will not return any value.

Versions 3.0 and 2.0 of the Open API are compatible, with the following main exceptions:

- in DOCX/RTF export parameters formatting options have been changed
- in HTML export parameters UseUnicode flag is replaced with EncodingType parameter
- in TXT/CSV/HTML export parameters CodePageType parameter is no longer supported
- WordXML export format is no longer supported

Below is the full list of changes:

<table>
<thead>
<tr>
<th>Object/Enumeration</th>
<th>Property/Method/Constant</th>
<th>What have happened?</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSVExportSettings</td>
<td>CodePageType</td>
<td>Removed</td>
<td>The property is no longer supported.</td>
</tr>
<tr>
<td></td>
<td>CodePageType</td>
<td>Removed</td>
<td>The property is no longer supported.</td>
</tr>
<tr>
<td></td>
<td>HTMLFormatMode</td>
<td>Removed</td>
<td>Similar functionality is provided via the <strong>AllowCss</strong> property.</td>
</tr>
<tr>
<td>HTMLExportSettings</td>
<td>SeparatePages</td>
<td>Removed</td>
<td>The property is no longer supported.</td>
</tr>
<tr>
<td></td>
<td>UseUnicode</td>
<td>Removed</td>
<td>Similar functionality is provided via the <strong>EncodingType</strong> property.</td>
</tr>
<tr>
<td>MSWordXmlExportSettings</td>
<td></td>
<td>Removed</td>
<td>WordXML export format is no longer supported.</td>
</tr>
<tr>
<td>OutputFileFormatEnum</td>
<td>OFF_MSWordXML</td>
<td>Removed</td>
<td>WordXML export format is no longer supported.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The Formatting options have been changed in DOCX/RTF export parameters.</td>
</tr>
<tr>
<td>RTFSynthesisModeEnum</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TextExportSettings</td>
<td>CodePageType</td>
<td>Removed</td>
<td>The property is no longer supported.</td>
</tr>
</tbody>
</table>
ABBYY Recognition Server 4 COM-based API
Working with Properties

The interfaces of COM-based API objects have various properties and methods. Visual Basic users will be familiar with the notion of property, therefore the text below explains how the properties are handled in C++.

For a C++ user, a property is a pair of methods (get and put for read-write properties) or a single get method (for read-only properties). However, the "Native COM support" in Microsoft Visual C++ makes the handling of properties more like in Visual Basic.

The Open API properties may be of the following types:

<table>
<thead>
<tr>
<th>Visual Basic type</th>
<th>C++ type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean (with two values, True and False)</td>
<td>VARIANT_BOOL (with two values VARIANT_TRUE and VARIANT_FALSE)</td>
</tr>
<tr>
<td>Long</td>
<td>long</td>
</tr>
<tr>
<td>Double</td>
<td>double</td>
</tr>
<tr>
<td>String</td>
<td>BSTR, a pointer to Unicode string. Zero value specifies an empty string.</td>
</tr>
<tr>
<td>Object</td>
<td>IUnknown - derived interface</td>
</tr>
<tr>
<td></td>
<td>Enumeration</td>
</tr>
</tbody>
</table>

See the details of working with different types of properties below:

- [Working with simple properties](#)
- [Working with string properties](#)
- [Working with object properties](#)
ABBYY Recognition Server 4 COM-based API
Important: The indexing of ABBYY Recognition Server Open API collections starts with 0.

Collection classes are implemented for several types of objects in the Open API. The following collection types are available:

- **StringsCollection** — a collection of strings
- **OutputFormatSettingsCollection** — a collection of `OutputFormatSettings` objects
- **InputFiles** — a collection of `InputFile` objects
- **OutputDocuments** — a collection of `OutputDocument` objects
- **Workflows** — a collection of `Workflow` objects
- **DocumentInfoItems** — a collection of `DocumentInfoItem` objects
- **DocumentAttributes** — a collection of `DocumentAttribute` objects
- **DocumentTypes** — a collection of `DocumentType` objects
- **IndexingFields** — a collection of `IndexingField` objects

These collections are independent objects and are used to pass various sets of parameters to functions that require them.

All these objects are characterized by a common set of properties and methods that make them collections.

These properties and methods are (in IDL notation):

**Standard collection-specific properties and methods:**

```
// This property stores the number of elements in the collection
HRESULT Count( [out, retval]long* pVal );
// This method provides access to a single collection element
HRESULT Item( [in]long index, [out, retval]<collection type>* pVal );
```

**ABBYY Recognition Server collection-specific properties and methods (optional):**

```
// Inserts a new element at the specified position
HRESULT Insert( [in]<collection type> newVal, [in]long index );
// Adds a new element at the end of the collection
HRESULT Add( [in]<collection type> newVal );
// Removes an element from the collection
HRESULT Remove( [in]long index );
// Removes all elements from the collection
HRESULT RemoveAll();
```

See also

See samples: Hello, Listening, AsyncProcessing, Sample for ASP.NET.
ABBYY Recognition Server 4 COM-based API
Working with Connectable Objects

The **Client** object in the ABBYY Recognition Server Open API is a so-called "connectable object". This means that it implements the **IConnectionPointContainer** interface. A connectable object ensures communication between the Open API and its clients.

This connectable object provides connection points of two types — one that uses a **dispatch** interface, and one that uses an interface derived from **IUnknown**. The **dispatch** interface is designed for automatic use in Visual Basic and similar environments, while the vtbl-based interface is suitable for use in C++.

An Open API client application that needs to receive notifications of certain events in the Open API must implement interfaces of a specific type and "advise" objects implementing these interfaces to the corresponding connectable objects.

In Visual Basic this is done by simply declaring the object **WithEvents** and implementing the corresponding methods of the callback interface. The procedure for Visual Basic is described in the **IClientEvents** interface.

Below follows an example of connecting an object on the client side.

```cpp
class CClientEventsListener : public IClientEvents {
public:
    ...

    // Provide simple implementation of IUnknown methods. They may also be
    // implemented through inheritance from some standard class with COM support
    ULONG AddRef();
    ULONG Release();
    HRESULT QueryInterface(REFIID iid, void** ppvObject) {
        if( ppvObject == 0 )
            return E_INVALIDARG;

        if( riid == __uuidof(IClientEvents) ) {
            *ppvObject = static_cast<IClientEvents*>( this );
        } else if( riid == IID_IUnknown ) {
            *ppvObject = static_cast<IUnknown*>( this );
        } else {
            *ppvObject = 0;
            return E_NOINTERFACE;
        }

        AddRef();
        return S_OK;
    }

    // Provide IClientEvents method implementation
    HRESULT OnJobComplete(BSTR, IXmlResult*);
};
```
Thus we have a `CClientEventsListener` class that may be used to receive notifications from the `Client` object. The following section of code advises this object to the notifications source (error handling is omitted):

```cpp
// Suppose that we have already created the Client object
IClient* client;
IConnectionPointContainer* pContainer=0;
client->QueryInterface(IID_IConnectionPointContainer, (void**)&pContainer);
IConnectionPoint* pPoint=0;
pContainer->FindConnectionPoint(__uuidof(IClientEvents),
    &pPoint);
CClientEventsListener listener;
IUnknown* listenerUnknown=0;
listener.QueryInterface(IID_IUnknown, (void**)&listenerUnknown);
// A variable to store the cookie returned from the IConnectionPoint::Advise method
DWORD cookie;
pPoint->Advise(listenerUnknown, &cookie);
...
// After notification, the listener is no longer needed and should be unadvised
pPoint->Unadvise(cookie);
```

Refer to documentation on COM for a more detailed description of connectable objects.

See also

- `Client`
- `IClientEvents`
ABBYY Recognition Server 4 COM-based API
# Standard Return Codes of ABBYY Recognition Server COM-based API Functions

Below is a list of the standard return codes of the ABBYY Recognition Server COM-based API functions and properties.

<table>
<thead>
<tr>
<th>Return code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>S_OK</td>
<td>Method completed successfully.</td>
</tr>
<tr>
<td>E_OUTOFMEMORY</td>
<td>There was not enough memory to perform the operation.</td>
</tr>
<tr>
<td>E_UNEXPECTED</td>
<td>Unexpected internal error.</td>
</tr>
<tr>
<td>E_NOTIMPL</td>
<td>Method is not implemented.</td>
</tr>
<tr>
<td>E_POINTER</td>
<td>Invalid pointer argument.</td>
</tr>
<tr>
<td>E_INVALIDARG</td>
<td>One or more arguments are invalid.</td>
</tr>
<tr>
<td>CO_E_OBJNOTCONNECTED</td>
<td>A pointer to an object was passed that is no longer valid (this object was destroyed).</td>
</tr>
<tr>
<td>E_FAIL</td>
<td>Unspecified error.</td>
</tr>
</tbody>
</table>

Other return codes are possible, specifically those related to file system errors.
Using the COM-based API within 64-bit Applications

The ABBYY Recognition Server setup program automatically configures the system to use COM API.

For troubleshooting purposes, see the steps required for manual configuration.

To create a COM+ application, do the following:

1. In the Component Services window (Start > Settings > Control Panel > Administrative Tools > Component Services), select Component Services > Computers > My Computer > COM+ Applications.

2. On the shortcut menu of the COM+ Applications item, select New > Application. In the COM+ Application Install Wizard, click Next > Create an empty application > Server application. Enter a name for the application. Click Next and specify the application identity, i.e. the same user account that is used for running the ABBYY Recognition Server service (Select Start > Control Panel > Administrative Tools > Services and see ABBYY Recognition Server 4 Server Manager service). A new application will be displayed in the tree on the left.

3. On the shortcut menu of the Components item of the newly created application, select New > Component. In the COM+ Component Install Wizard, click Next > Import components that are already registered. Select the 32-Bit Registry item and choose all the ABBYYOCRServer.XXX components from the list of registered COM components.

4. For each component in the list, disable the Enforce component level access checks option. To achieve this, go to the component Properties dialog box and click the Security tab.

5. Each user that is supposed to use the COM API should be allowed to use the COM+ application.

To grant the necessary permissions, do the following:

- Select Start > Control Panel > Administrative Tools > Component Services.
- Find the configured COM+ application.
- Open the list of roles and add a role. You can use any name for the role, for example "APIUsers."
- Finally, add the desired user to this role. To allow any user to use the COM API, add the user "Everyone."

6. Register OCRServerClient.dll:

   - Select Start > Run > cmd.
Enter "regsvr32" and the path to the OCRServerClient.dll, for example: regsvr32 "C:\Program Files (x86)\ABBYY Recognition Server 4.0\Bin\OCRServerClient.dll"

After the COM+ application is created, the COM-based API can be used within 64-bit project with the following limitation: You must not use objects created with the help of the operator new and the CoCreateInstance method as the input parameters of the methods of COM-based API objects. Instead, use:

- the IClient::CreateXmlTicket method for creating an XmlTicket object
- the IClient::CreateOutputFormatSettings method for creating objects derived from an OutputFormatSettings object
ABBY Recognition Server 4 COM-based API
Distribution of Applications Using the ABBYY Recognition Server COM-based API Library

The COM-based API component must be installed on a workstation if you want to run applications which use the ABBYY Recognition Server COM-based API library. You can install the COM-based API by using an installation program which automates the setup process (see the System Administrator's Guide for details).

If you want to install the COM-based API component manually, please do the following:

1. Copy the files marked as obligatory in the table provided in the ABBYY Recognition Server COM-based API Distribution Kit section. Note: All the files must be copied into the same folder, unless otherwise stated in the description of the file.
3. Register OCRServerClient.dll in the Windows registration database:

   regsvr32.exe OCRServerClient.dll.

   Note: Before uninstalling, you must unregister the OCRServerClient.dll:

   regsvr32.exe -u OCRServerClient.dll

Important! Never redistribute the ABBYY Recognition Server COM-based API type library and files with API descriptions (OCRServerClient.h, OCRServerClient.tlb, OCRServerClient.th, OCRServerClient.tli, OCRServerClient_i.c).

See also

ABBYY Recognition Server COM-based API Distribution Kit
ABBYY Recognition Server 4 COM-based API
ABBYY Recognition Server COM-based API Distribution Kit

The ABBYY Recognition Server COM-based API library is implemented as a set of dynamic link libraries (DLL) and additional modules. After you have installed the library, its type library is registered in the system registry.

The description of the files of the library is given in the table below. Some paths are given relative to the root folder of the ABBYY Recognition Server installation. The root folder is set up when installing the ABBYY Recognition Server COM-based API. This table also specifies what files must be distributed as part of your application, and what files should not.

<table>
<thead>
<tr>
<th>File or folder</th>
<th>Description</th>
<th>Distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>%ALLUSERSPROFILE%\Application Data\ABBYY Recognition Server 4.0\Inc*</td>
<td>OCRServerClient.h, OCRServerClient.tlb, OCRServerClient_i.c, Microsoft.VC90.CRT</td>
<td>ABBYY Recognition Server COM-based API type library description and API declaration files.</td>
</tr>
</tbody>
</table>

The root folder

| Help | RecognitionServer0.chm | This manual. | No. |
| Bin | FineObj.dll, LangInfo.dll, LangInfoUnicode.dll, msvcr90.dll | ABBYY Recognition Server COM-based API system modules. | Mandatory. |
| OCRServerClient.dll | ABBYY Recognition Server COM-based API system module. | Mandatory. |
| OCRServerClient0.dll | ABBYY Recognition Server COM-based API resource module for the English interface. The schemas in XML Schema definition language (XSD) used to describe the XML ticket and XML result files. | Mandatory. |
| XmlTicket.xsd, XmlResult.xsd | No. |

* — In Microsoft Windows Vista, Windows 7, Windows 8, and Windows Server 2008 OS this folder is installed in the %PUBLIC%\ABBYY\ABBYY Recognition Server 4.0\Inc folder.

See also

Distribution of Applications Using the ABBYY Recognition Server COM-based API Library
**Client Object (IClient Interface)**

This object allows you to connect to ABBYY Recognition Server and to process recognition jobs in synchronous and asynchronous modes. It can be created by using the **CreateObject** (Visual Basic) or **CoCreateInstance** (C++/C) methods.

The **Client** object is a so-called "connectable object" (see details in the **Connectable Objects** section). It may be declared **WithEvents** in Visual Basic. For C++ user this fact means that it supports the **IConnectionPointContainer** interface. To receive notification events during processing, a C++ user should create an object derived from the **IClientEvents** interface, then set up the connection between it and events source implemented in the **Client** object by standard COM means.

### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ServerExceptionsFolder</strong></td>
<td><strong>String</strong>, read-only</td>
<td>Returns the path to the Exceptions folder on the Processing Server.</td>
</tr>
<tr>
<td><strong>Workflows</strong></td>
<td><strong>StringsCollection</strong>, read-only</td>
<td>Returns a collection of available workflows. Only workflows with the following types of the Input folder are listed: Shared Folder, FTP Folder.</td>
</tr>
<tr>
<td><strong>WorkflowsSettings</strong></td>
<td><strong>Workflows</strong>, read-only</td>
<td>Returns a collection of workflow settings for all available workflows. Only workflows with the following types of the Input folder are listed: Shared Folder, FTP Folder.</td>
</tr>
</tbody>
</table>

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Connect</strong></td>
<td>Establishes a connection with the server.</td>
</tr>
<tr>
<td><strong>CreateDocumentAttribute</strong></td>
<td>Creates a <strong>DocumentAttribute</strong> object.</td>
</tr>
<tr>
<td><strong>CreateInputFile</strong></td>
<td>Creates a <strong>InputFile</strong> object.</td>
</tr>
<tr>
<td><strong>CreateOutputFormatSettings</strong></td>
<td>Creates an <strong>OutputFormatSettings</strong> object.</td>
</tr>
<tr>
<td><strong>CreateXmlTicket</strong></td>
<td>Creates an <strong>XmlTicket</strong> object based on the specified workflow.</td>
</tr>
<tr>
<td><strong>DeleteJob</strong></td>
<td>Completely deletes the job.</td>
</tr>
<tr>
<td><strong>IsListening</strong></td>
<td>Allows you to learn whether receipt of job completion notifications from a selected workflow is enabled or disabled.</td>
</tr>
<tr>
<td><strong>GetJobState</strong></td>
<td>Allows you to get information about current state of the specified job.</td>
</tr>
<tr>
<td><strong>ProcessFile</strong></td>
<td>Uses the parameters of the specified workflow to recognize the specified input image.</td>
</tr>
</tbody>
</table>
**ProcessFileAsync**
Uses the parameters of the specified workflow to recognize the input image in asynchronous mode.

**ProcessXmlTicket**
Uses the parameters of Xml Ticket and the specified workflow to recognize the input image.

**ProcessXmlTicketAsync**
Uses the parameters of Xml Ticket and the specified workflow to recognize the input image in asynchronous mode.

**StartListening**
Allows you to get notifications about the completion of work by a specified workflow.

**UpdateWorkflow**
Changes settings of the specified workflow

See also

**XmlTicket**, **StringsCollection**.

See samples: [Hello], [Listening], [AsyncProcessing], [Sample for ASP.NET].
ABBYY Recognition Server 4 COM-based API
IClientEvents Interface

This is a callback interface that is used for reporting events from the Client object to the listeners. This interface is implemented on the client side. As it derives from the IUnknown interface, the client object should also implement the IUnknown methods. This interface is designed primarily for use in C++. Visual Basic users that wish to receive notifications from the Client object should declare it WithEvents and implement the following Subs:

```
Public WithEvents cl As ABBYYRecognitionServer.Client

Private Sub cl_OnJobComplete(ByVal jobId As String, ByVal result As XmlResult)
...
End Sub
```

**Note:** You cannot call the ProcessFile, ProcessFileAsync, ProcessXmlTicket, and ProcessXmlTicketAsync methods in the event handler.

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>OnJobComplete</strong></td>
<td>Provides information about job completion to the client.</td>
</tr>
</tbody>
</table>

### Remarks

To receive notifications correctly, please make sure that your application releases control and runs the Windows message loop. See the sample below.

### Visual Basic 6.0

```
Dim WithEvents Client As ABBYYRecognitionServer.Client
Dim Workflow As String
Dim Filename As String

Private Sub StartListening_Click()
    ' The StartListening method is called and then control is released
    Client.StartListening Workflow, True
End Sub

Private Sub ProcessAsync_Click()
    ' The ProcessFileAsync method is called and then control is released
    Client.ProcessFileAsync Workflow, Filename
End Sub
```
Private Sub client_OnJobComplete(ByVal jobId As String, ByVal result As XmlResult)
    ' Here is a reaction to job completion
End Sub

See also

Client,
Working with Connectable Objects
ABBY Recognition Server 4 COM-based API
XmlTicket Object (IXmlTicket Interface)

This object represents the processing parameters of one job. It can be created by using the CreateXmlTicket method of the Client object based on one workflow from the list in the IClient::Workflows property. Alternatively, it can be created by using the CreateObject (Visual Basic) or CoCreateInstance (C++/C) methods.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attributes</td>
<td>DocumentAttributes, read-only</td>
<td>Document attributes.</td>
</tr>
<tr>
<td>ExportParams</td>
<td>ExportParams, read-only</td>
<td>Returns a reference to the ExportParams object, where you can set export parameters.</td>
</tr>
<tr>
<td>Id</td>
<td>String, read-only</td>
<td>Returns a unique ID.</td>
</tr>
<tr>
<td>InputFiles</td>
<td>InputFiles, read-only</td>
<td>Returns a reference to the image collection.</td>
</tr>
<tr>
<td>Name</td>
<td>String</td>
<td>Stores the name of XML Ticket, which will be used when saving the results. The default value is an empty string.</td>
</tr>
<tr>
<td>OcrTimeout</td>
<td>Long</td>
<td>Specifies a recognition time limit for a job in milliseconds. If the job processing time exceeds the limit, the job will be discarded. The value of this property must be nonnegative. If you set the value of this property to 0, no time limit is used. The default value of this property is 0.</td>
</tr>
<tr>
<td>PreprocessingParams</td>
<td>PreprocessingParams, read-only</td>
<td>Returns a reference to the PreprocessingParams object, where you can set image processing parameters.</td>
</tr>
<tr>
<td>Priority</td>
<td>PriorityEnum</td>
<td>Sets the priority of the job. The default value is P_Normal.</td>
</tr>
<tr>
<td>RecognitionParams</td>
<td>RecognitionParams, read-only</td>
<td>Returns a reference to the RecognitionParams object, where you can set recognition parameters. Stores any user-defined string. The string is passed to the UserProperty property of the XmlResult object. The default value is empty string.</td>
</tr>
<tr>
<td>UserProperty</td>
<td>String</td>
<td></td>
</tr>
</tbody>
</table>

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AddImage</td>
<td>Adds the image to XmlTicket.</td>
</tr>
</tbody>
</table>
**LoadFromFile** Restores the contents of the object from a file on disk.

**SaveToFile** Saves the contents of the object into a file on disk.

See also

**XmlResult**, **XML Ticket description**.

See sample: [Sample for ASP.NET](#).
ABBYY Recognition Server 4 COM-based API
# XmlResult Object (IXmlResult Interface)

This object represents the parameters and results of processing one job.

## Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Errors</td>
<td><code>StringsCollection</code>, read-only</td>
<td>Returns a list of errors that occurred when processing the job.</td>
</tr>
<tr>
<td>ExportParams</td>
<td><code>ExportParams</code>, read-only</td>
<td>Returns the export parameters</td>
</tr>
<tr>
<td>Id</td>
<td><code>String</code>, read-only</td>
<td>Returns the unique ID of the job.</td>
</tr>
<tr>
<td>InputFiles</td>
<td><code>InputFiles</code>, read-only</td>
<td>Returns the collection of image files and processing results.</td>
</tr>
<tr>
<td>JobDocuments</td>
<td><code>JobDocuments</code>, read-only</td>
<td>Returns the collection of job documents.</td>
</tr>
<tr>
<td>IsFailed</td>
<td><code>Boolean</code>, read-only</td>
<td>If an error occurred during processing, this property will be set to TRUE.</td>
</tr>
<tr>
<td>LoadedFrom</td>
<td><code>String</code>, read-only</td>
<td>Returns the path to the file from which the object was initialized. The path may be empty if the file was deleted by Open API, e.g. if the file should not be published in compliance with the settings.</td>
</tr>
<tr>
<td>Name</td>
<td><code>String</code>, read-only</td>
<td>Returns the name of XML Result, which coincides with the name of the XML Ticket that was used for recognition.</td>
</tr>
<tr>
<td>OcrTimeout</td>
<td><code>Long</code>, read-only</td>
<td>Returns the recognition time limit for a job in milliseconds. If the job processing time exceeds the limit, the job is discarded. If the value of this property is 0, no time limit was used.</td>
</tr>
<tr>
<td>PreprocessingParams</td>
<td><code>PreprocessingParams</code>, read-only</td>
<td>Returns the image processing parameters.</td>
</tr>
<tr>
<td>Priority</td>
<td><code>PriorityEnum</code>, read-only</td>
<td>Returns the priority of the job.</td>
</tr>
<tr>
<td>RecognitionParams</td>
<td><code>RecognitionParams</code>, read-only</td>
<td>Returns the recognition parameters.</td>
</tr>
<tr>
<td>Statistics</td>
<td><code>Statistics</code>, read-only</td>
<td>Returns the job processing statistics.</td>
</tr>
<tr>
<td>UserProperty</td>
<td><code>String</code>, read-only</td>
<td>Returns the user-defined string which is specified the <code>UserProperty</code> property of the <code>XmlTicket</code> object.</td>
</tr>
<tr>
<td>Warnings</td>
<td><code>StringsCollection</code>, read-only</td>
<td>Returns a list of warnings that were issued when processing the job.</td>
</tr>
</tbody>
</table>
### BarcodeText

**String**, read-only

The property is retained for backward compatibility. For getting the text of the separation barcode, use the BarcodeText property of the JobDocument object.

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>LoadFromFile</strong></td>
<td>Restores the contents of the object from a file on disk.</td>
</tr>
<tr>
<td><strong>SaveToFile</strong></td>
<td>Saves the contents of the object into an XML file on disk. Note: If a folder for XML Result file is specified in the Remote Administration Console (on the Output tab of the Workflow Properties dialog box), an XML file will be saved to this folder and the folder which is specified in this method will be ignored.</td>
</tr>
</tbody>
</table>

### See also

XmlTicket, XML Result description.

### See samples: Hello, Sample for ASP.NET.
ABBYY Recognition Server 4 COM-based API
InputFiles Object

This object represents a collection of InputFile objects.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count</td>
<td>Long, read-only</td>
<td>Stores the number of elements in the collection.</td>
</tr>
</tbody>
</table>

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add</td>
<td>Adds a new element at the end of the collection.</td>
</tr>
<tr>
<td>Item</td>
<td>Provides access to a single element of the collection.</td>
</tr>
<tr>
<td>Remove</td>
<td>Removes an element from the collection.</td>
</tr>
<tr>
<td>RemoveAll</td>
<td>Removes all the elements from the collection.</td>
</tr>
</tbody>
</table>

See also

InputFile, XmlTicket, XmlResult, Working with Collections.

See sample: Sample for ASP.NET.
ABBYY Recognition Server 4 COM-based API
InputFile Object (IInputFile Interface)

This object represents one input image file and the results of processing this file.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attributes</td>
<td>DocumentAttributes, read-only</td>
<td>The property is retained for backward compatibility. For getting the document attributes, use the Attributes property of the JobDocument object.</td>
</tr>
<tr>
<td>CustomText</td>
<td>String, read-only</td>
<td>The property is retained for backward compatibility. For getting the custom text, use the CustomText property of the JobDocument object.</td>
</tr>
<tr>
<td>Errors</td>
<td>StringsCollection, read-only</td>
<td>Returns a list of errors that occurred when processing the image.</td>
</tr>
<tr>
<td>FileName</td>
<td>String</td>
<td>Stores the name of the image file.</td>
</tr>
<tr>
<td>Note: You cannot include any subfolders into the file name.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ID</td>
<td>String, read-only</td>
<td>Stores the unique identifier of the input file.</td>
</tr>
<tr>
<td>OutputDocuments</td>
<td>OutputDocuments, read-only</td>
<td>Returns a collection of output files of different formats belonging to the document.</td>
</tr>
<tr>
<td>Pages</td>
<td>Pages, read-only</td>
<td>Returns a collection of pages in the input image file.</td>
</tr>
<tr>
<td>Password</td>
<td>String</td>
<td>Stores the password for accessing the PDF file. The default value is an empty string.</td>
</tr>
<tr>
<td>Statistics</td>
<td>Statistics, read-only</td>
<td>Returns image processing statistics.</td>
</tr>
<tr>
<td>Warnings</td>
<td>StringsCollection, read-only</td>
<td>Returns a list of warnings that were issued when processing the image.</td>
</tr>
</tbody>
</table>

See also

InputFiles.

See sample: Sample for ASP.NET.
ABYY Recognition Server 4 COM-based API
JobDocuments Object

This object represents a collection of JobDocument objects.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count</td>
<td>Long, read-only</td>
<td>Stores the number of elements in the collection.</td>
</tr>
</tbody>
</table>

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add</td>
<td>Adds a new element at the end of the collection.</td>
</tr>
<tr>
<td>Item</td>
<td>Provides access to a single element of the collection.</td>
</tr>
<tr>
<td>Remove</td>
<td>Removes an element from the collection.</td>
</tr>
<tr>
<td>RemoveAll</td>
<td>Removes all the elements from the collection.</td>
</tr>
</tbody>
</table>

See also

JobDocument, XmlTicket, XmlResult, Working with Collections.
ABBYY Recognition Server 4 COM-based API
JobDocument Object (IJobDocument Interface)

This object represents one input image file and the results of processing this file.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attributes</td>
<td>DocumentAttributes, read-only</td>
<td>Stores document attributes.</td>
</tr>
<tr>
<td>BarcodeText</td>
<td>String, read-only</td>
<td>Returns the text of the separation barcode.</td>
</tr>
<tr>
<td>CustomText</td>
<td>String, read-only</td>
<td>Returns a list of errors that occurred when processing the document.</td>
</tr>
<tr>
<td>Errors</td>
<td>StringsCollection, read-only</td>
<td>Returns a list of warnings that were issued when processing the document.</td>
</tr>
<tr>
<td>Name</td>
<td>String</td>
<td>Stores the name of the job document. The name is generated automatically.</td>
</tr>
<tr>
<td>OutputDocuments</td>
<td>OutputDocuments, read-only</td>
<td>Returns a collection of output files of different formats belonging to the document.</td>
</tr>
<tr>
<td>PagePositions</td>
<td>PagePositions, read-only</td>
<td>Returns a collection of the output document positions.</td>
</tr>
<tr>
<td>Statistics</td>
<td>Statistics, read-only</td>
<td>Returns document processing statistics.</td>
</tr>
<tr>
<td>Warnings</td>
<td>StringsCollection, read-only</td>
<td>Returns a list of warnings that were issued when processing the document.</td>
</tr>
</tbody>
</table>
ABBYY Recognition Server 4 COM-based API
Workflows Object (IWorkflows Interface)

This object provides access to a collection of Workflow objects which represent the workflow settings. A reference to this object is returned by the IClient::WorkflowsSettings property.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count</td>
<td>Long</td>
<td>Stores the number of elements in the collection.</td>
</tr>
</tbody>
</table>

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Item</td>
<td>Provides access to a single element of the collection.</td>
</tr>
</tbody>
</table>

See also

- Working with Collections
- Workflow
- IClient::WorkflowsSettings
ABBY Recognition Server 4 COM-based API
Workflow Object (IWorkflow Interface)

This object represents the workflow settings: the workflow name, the paths to the Input folder and the Exception folder, and the export parameters. All the properties are read-only.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ExceptionsFolder</td>
<td>String, read-only</td>
<td>Returns the path to the Exceptions Folder of the workflow.</td>
</tr>
<tr>
<td>ExportParams</td>
<td>ExportParams, read-only</td>
<td>Returns the export parameters.</td>
</tr>
<tr>
<td>IndexingSettings</td>
<td>IndexingSettings, read-only</td>
<td>Returns the indexing settings.</td>
</tr>
<tr>
<td>InputFolder</td>
<td>String, read-only</td>
<td>Returns the path to the Input Folder of the workflow.</td>
</tr>
<tr>
<td>Name</td>
<td>String, read-only</td>
<td>Returns the name of the workflow.</td>
</tr>
<tr>
<td>WorkflowState</td>
<td>WorkflowStateEnum, read-only</td>
<td>Returns a collection of workflow states.</td>
</tr>
</tbody>
</table>

See also

Workflows
ABBYY Recognition Server 4 COM-based API
PreprocessingParams Object
(IPreprocessingParams Interface)

This object specifies how an image will be preprocessed before analysis and recognition.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConvertToBWFormat</td>
<td>Boolean</td>
<td>Specifies if the image must be converted to black and white during preprocessing. The default value is FALSE.</td>
</tr>
<tr>
<td>CorrectResolution</td>
<td>Boolean</td>
<td>Specifies if the image resolution must be corrected. The default value is TRUE.</td>
</tr>
<tr>
<td>Deskew</td>
<td>Boolean</td>
<td>Specifies if the skew angle for an image must be corrected during preprocessing. This mode is recommended if you want to automatically correct skew for images you work with. The default value is TRUE.</td>
</tr>
<tr>
<td>RemoveGarbage</td>
<td>Boolean</td>
<td>Specifies if garbage (excess dots that are smaller than a certain size) must be removed from the image during preprocessing. The default value is TRUE.</td>
</tr>
<tr>
<td>RemoveTexture</td>
<td>Boolean</td>
<td>Specifies if background noise must be cleared before the recognition process starts. The default value is TRUE.</td>
</tr>
<tr>
<td>RotationType</td>
<td>RotationTypeEnum</td>
<td>Specifies what type of rotation will be performed upon the image during preprocessing. The default value is RT_Automatic, which means that rotation will be detected automatically.</td>
</tr>
<tr>
<td>SplitDualPages</td>
<td>Boolean</td>
<td>Specifies if the dual pages must be split during preprocessing. The default value is FALSE.</td>
</tr>
</tbody>
</table>

See also

XmlTicket, XmlResult.

See sample: Sample for ASP.NET.
RecognitionParams Object (IRecognitionParams Interface)

This object allows you to tune the recognition parameters.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Languages</td>
<td>StringsCollection, read-only</td>
<td>Specifies a collection of recognition languages. Elements of this collection must be internal names from the first column of the table of the ABBYY Recognition Server recognition languages. The default value is &quot;English&quot;.</td>
</tr>
<tr>
<td>LookForBarcodes</td>
<td>Boolean</td>
<td>Specifies whether barcodes must be recognized. The default value is TRUE.</td>
</tr>
<tr>
<td>ProhibitHiddenTextDetection</td>
<td>Boolean</td>
<td>This attribute is set to TRUE by default, in which case text on pictures in input files will not be recognized. If it is set to FALSE, the program will recognize text in picture areas and create a text layer for these areas in the output document.</td>
</tr>
<tr>
<td>ProhibitPictureDetection</td>
<td>Boolean</td>
<td>Prohibits picture area detection. The default value is FALSE.</td>
</tr>
<tr>
<td>ProhibitTableDetection</td>
<td>Boolean</td>
<td>Prohibits table area detection. The default value is FALSE.</td>
</tr>
<tr>
<td>RecognitionMode</td>
<td>RecognitionModeEnum</td>
<td>Specifies the recognition mode. The default value is RM_FullPage.</td>
</tr>
<tr>
<td>RecognitionQuality</td>
<td>RecognitionQualitySettingEnum</td>
<td>Specifies the mode which optimizes recognition for quality or speed. The default value is RQS_Thorough, which corresponds to full mode.</td>
</tr>
</tbody>
</table>

The value of this property is an OR superposition of the TextTypeEnum. The value of this...
TextTypes Long

property must not be an empty set. The default value is TT_Normal, which corresponds to common typographic text.

VerificationMode VerificationModeEnum

Controls the verification parameters.

Specifies the verification threshold, i.e. the maximum percentage of low-confidence characters. If this percentage is exceeded for a document, the document will be submitted to the verification stage.

See also

XmlTicket, XmlResult.

See sample: Sample for ASP.NET.
ABBYY Recognition Server 4 COM-based API
DocumentAttributes Object

This object represents a collection of DocumentAttribute objects.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count</td>
<td>Long, read-only</td>
<td>Stores the number of elements in the collection.</td>
</tr>
<tr>
<td>DocumentType</td>
<td>String</td>
<td>Specifies a document type.</td>
</tr>
<tr>
<td>SkipManualIndexing</td>
<td>Boolean</td>
<td>Specifies whether manual indexing is to be skipped. If the value of this property is FALSE, the document will be queued for processing at the Indexing Station.</td>
</tr>
</tbody>
</table>

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add</td>
<td>Adds a new element at the end of the collection.</td>
</tr>
<tr>
<td>Item</td>
<td>Provides access to a single element of the collection.</td>
</tr>
<tr>
<td>Remove</td>
<td>Removes an element from the collection.</td>
</tr>
<tr>
<td>RemoveAll</td>
<td>Removes all the elements from the collection.</td>
</tr>
</tbody>
</table>

See also

XmlTicket,
DocumentAttribute
ABBYY Recognition Server 4 COM-based API
DocumentAttribute Object
(IDocumentAttribute Interface)

This object represents a document attribute. It may be typecast to one of its child objects:

- **BooleanAttribute**
- **EnumerationAttribute**
- **SingleLineAttribute**
- **MultipleLinesAttribute**
- **RegularExpressionAttribute**

These objects allow access to values of different types and inherit the properties of the **DocumentAttribute** object. They are also elements of the **DocumentAttributes** collection.

**Properties**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>String, read-only</td>
<td>Attribute name.</td>
</tr>
<tr>
<td>Type</td>
<td>AttributeTypeEnum,</td>
<td>read-only Attribute type.</td>
</tr>
</tbody>
</table>

See also

**DocumentAttributes**
BooleanAttribute Object (IBooleanAttribute Interface)

This object provides access to the value of the Checkbox attribute. The IBooleanAttribute interface is a child object of the IDocumentAttribute interface and inherits its properties. It is an element of the DocumentAttributes collection.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>Boolean</td>
<td>This property sets the value of the Checkbox attribute.</td>
</tr>
</tbody>
</table>

See also

DocumentAttribute, DocumentAttributes, AttributeTypeEnum
EnumerationAttribute Object
(IEnumerationAttribute Interface)

This object provides access to the value of the List attribute. The IEnumerationAttribute interface is a child object of the IDocumentAttribute interface and inherits its properties. It is an element of the DocumentAttributes collection.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>String</td>
<td>This property specifies an element of the list.</td>
</tr>
</tbody>
</table>

See also

DocumentAttribute, DocumentAttributes, AttributeTypeEnum
ABBYY Recognition Server 4 COM-based API
SingleLineAttribute Object
(ISingleLineAttribute Interface)

This object provides access to the value of the Single line attribute. The ISingleLineAttribute interface is a child object of the IDocumentAttribute interface and inherits its properties. It is an element of the DocumentAttributes collection.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>String</td>
<td>This property contains a single line of text as a value of the attribute.</td>
</tr>
</tbody>
</table>

See also

DocumentAttribute, DocumentAttributes, AttributeTypeEnum
MultipleLinesAttribute Object (IMultipleLinesAttribute Interface)

This object provides access to the value of the Multiple lines attribute. The IMultipleLinesAttribute interface is a child object of the IDocumentAttribute interface and inherits its properties. It is an element of the DocumentAttributes collection.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>String</td>
<td>This property contains multiple lines of text as a value of the attribute.</td>
</tr>
</tbody>
</table>

See also

DocumentAttribute, DocumentAttributes, AttributeTypeEnum
RegularExpressionAttribute Object (IRegularExpressionAttribute Interface)

This object provides access to the value of the Regular expression attribute. The IRegularExpressionAttribute interface is a child object of the IDocumentAttribute interface and inherits its properties. It is an element of the DocumentAttributes collection.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>String</td>
<td>This property contains a regular expression as a value of the attribute.</td>
</tr>
</tbody>
</table>

See also

DocumentAttribute, DocumentAttributes, AttributeTypeEnum
ABBYY Recognition Server 4 COM-based API
IndexingSettings Object (IIndexingSettings Interface)

This object provides functionality for tuning indexing settings.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DefaultDocumentType</td>
<td>String</td>
<td>Specifies the default document type name.</td>
</tr>
<tr>
<td>DocumentTypes</td>
<td>DocumentTypes, read-only</td>
<td>Stores a collection of document types.</td>
</tr>
</tbody>
</table>

See also

DocumentTypes
ABBYY Recognition Server 4 COM-based API
DocumentTypes Object

This object represents a collection of DocumentType objects.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count</td>
<td>Long</td>
<td>Stores the number of elements in the collection.</td>
</tr>
</tbody>
</table>

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add</td>
<td>Adds a new element at the end of the collection.</td>
</tr>
<tr>
<td>CreateNew</td>
<td>Creates a new DocumentType object and returns a reference to it.</td>
</tr>
<tr>
<td>Item</td>
<td>Provides access to a single element of the collection.</td>
</tr>
<tr>
<td>Remove</td>
<td>Removes an element from the collection.</td>
</tr>
<tr>
<td>RemoveAll</td>
<td>Removes all the elements from the collection.</td>
</tr>
</tbody>
</table>

See also

DocumentType, Working with Collections.
ABBYY Recognition Server 4 COM-based API
DocumentType Object (IDocumentType Interface)

This object represents document type for indexing.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>String, read-only</td>
<td>Returns the name of the document type.</td>
</tr>
<tr>
<td>Fields</td>
<td>IndexingFields, read-only</td>
<td>Returns a collection of indexing fields.</td>
</tr>
</tbody>
</table>

See also

IndexingFields.
ABYY Recognition Server 4 COM-based API
IndexingFields Object

This object represents a collection of `IndexingField` objects.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count</td>
<td>Long</td>
<td>Stores the number of elements in the collection.</td>
</tr>
</tbody>
</table>

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add</td>
<td>Adds a new element at the end of the collection.</td>
</tr>
<tr>
<td>CreateNew</td>
<td>Creates a new <code>IndexingField</code> object and returns a reference to it.</td>
</tr>
<tr>
<td>Item</td>
<td>Provides access to a single element of the collection.</td>
</tr>
<tr>
<td>Remove</td>
<td>Removes an element from the collection.</td>
</tr>
<tr>
<td>RemoveAll</td>
<td>Removes all the elements from the collection.</td>
</tr>
</tbody>
</table>

See also

- `IndexingField`
- `Working with Collections`
ABBYY Recognition Server 4 COM-based API
IndexingField Object (IIndexingField Interface)

This object represents an indexing field. It is a common interface for interfaces of indexing fields of different types:

- BooleanIndexingField
- EnumerationIndexingField
- MultipleLinesIndexingField
- RegularExpressionIndexingField
- SingleLineIndexingField

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>String, read-only</td>
<td>Returns the indexing field name.</td>
</tr>
<tr>
<td>Type</td>
<td>Attribute Type Enum, read-only</td>
<td>Returns the indexing field type.</td>
</tr>
<tr>
<td>IsRequired</td>
<td>Boolean, read-only</td>
<td>If the indexing field is required, this property returns TRUE.</td>
</tr>
</tbody>
</table>

All properties are set when the IIndexingFields::CreateNew method is called.

See also

IndexingFields.
ABBYY Recognition Server 4 COM-based API
BooleanIndexingField Object
(IBooleanIndexingField Interface)

This object represents an indexing field of boolean type. It inherits from IIndexingField interface and has the same properties.

See also

IndexingField,
IndexingFields.
EnumerationIndexingField Object (IEnumerationIndexingField Interface)

This object represents an indexing field of enumeration type. It inherits from \texttt{IndexingField} interface and has the same properties.

\textbf{Additional Properties}

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PossibleValues</td>
<td>\texttt{StringsCollection}, read-only</td>
<td>Returns list of possible values of the indexing field.</td>
</tr>
</tbody>
</table>

See also

\texttt{IndexingField}, \texttt{IndexingFields}.
ABBYY Recognition Server 4 COM-based API
SingleLineIndexingField Object
(ISingleLineIndexingField Interface)

This object represents an indexing field of single line. It inherits from IndexingField interface and has the same properties.

See also

IndexingField, IndexingFields.
ABBYY Recognition Server 4 COM-based API
MultipleLineIndexingField Object
(IMultipleLineIndexingField Interface)

This object represents an indexing field of multiple lines. It inherits from `IIndexingField` interface and has the same properties.

See also

[IndexingField],
[IndexingFields].
ABBYY Recognition Server 4 COM-based API
This object represents an indexing field of Regular Expression type. It inherits from `IIndexingField` interface and has the same properties.

**Additional Properties**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expression</td>
<td>String</td>
<td>Regular expression for the indexing field.</td>
</tr>
</tbody>
</table>

See also

- `IIndexingField`
- `IndexingFields`
ExportParams Object (IExportParams Interface)

This object provides functionality for tuning of export parameters of recognized text.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OutputFormats</td>
<td>OutputFormatSettingsCollection, read-only</td>
<td>Stores a collection of export parameters.</td>
</tr>
<tr>
<td>SeparationMethod</td>
<td>DocumentSeparationMethodEnum</td>
<td>Specifies the document separation method during export.</td>
</tr>
<tr>
<td>XmlResultLocation</td>
<td>String, read-only</td>
<td>Stores a location of an XML result file.</td>
</tr>
<tr>
<td>BarcodeType</td>
<td>BarcodeTypeEnum</td>
<td>Specifies the barcode type used for document separation.</td>
</tr>
</tbody>
</table>

See also

OutputFormatSettingsCollection,
OutputFormatSettings,
IXMLTicket::ExportParams,
IXMLResult::ExportParams
OutputDocuments Object

This object provides access to a collection of OutputDocument objects which represent output files of different formats belonging to the document. A reference to this object is returned by the IInputFile::OutputDocuments property.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count</td>
<td>Long,</td>
<td>read-only Stores the number of elements in the collection.</td>
</tr>
</tbody>
</table>

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Item</td>
<td>Provides access to a single element of the collection.</td>
</tr>
</tbody>
</table>

See also

Working with Collections, OutputDocument, InputFile
ABBY Recognition Server 4 COM-based API
OutputDocument Object (IOutputDocument Interface)

This object represents parameters of files of a particular format belonging to the document.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FileNames</td>
<td>StringsCollection, read-only</td>
<td>Stores a collection of paths to the output files.</td>
</tr>
<tr>
<td>FormatSettings</td>
<td>OutputFormatSettings, read-only</td>
<td>Stores export parameters.</td>
</tr>
</tbody>
</table>

See also

XmlTicket,
XmlNode
ABBYY Recognition Server 4 COM-based API
OutputFormatSettingsCollection Object (IOutputFormatSettingsCollection Interface)

This object provides access to a collection of OutputFormatSettings objects.

Properties

Name | Type | Description
--- | --- | ---
Count | Long, read-only | Stores the number of elements in the collection.

Methods

Name | Description
--- | ---
Add | Adds a new element at the end of the collection.
Item | Provides access to a single element of the collection.
Remove | Removes an element from the collection.
RemoveAll | Removes all the elements from the collection.

See also

OutputFormatSettings, ExportParams, Working with Collections.

See samples: Hello, Listening, AsyncProcessing, Sample for ASP.NET.
OutputFormatSettings Object (IOutputFormatSettings Interface)

This object provides functionality for tuning the export parameters of recognized text. It may be typecast to one of its child objects:

- MSWordExportSettings
- RTFExportSettings
- DOCXExportSettings
- XLExportSettings
- XLSXExportSettings
- XMLEXportSettings
- PDFExportSettings
- PDFAExportSettings
- EPUBExportSettings
- TextExportSettings
- CSVExportSettings
- HTMLExportSettings
- TiffExportSettings
- JpegExportSettings
- Jpeg2kExportSettings
- JBig2ExportSettings
- AltoExportSettings
- NoConversionExportSettings (does not provide any settings)
- InternalFormatExportSettings (does not provide any settings)

These objects allow access to export parameters of different formats and inherit all the properties of the OutputFormatSettings object. They are also elements of the OutputFormatSettingsCollection collection.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>OutputFileFormatEnum, read-only</td>
<td>Specifies the output file format.</td>
</tr>
<tr>
<td>FileFormat</td>
<td>Specifications the file naming and output path rule. The rule can contain the elements from the list below in angle brackets &lt;&gt; . The other elements must be typed without angle brackets. The elements</td>
<td></td>
</tr>
</tbody>
</table>
NamingRule String without angle brackets will be passed as is. Only one slash is allowed to specify a subfolder in the Output Folder.

The list of file naming and output path elements which can be added to the rule in angle brackets:

- **Dd** Input image day of month
- **Ext** Output format extension
- **FileName** Input image name
- **Folder** Input image folder (if available in the Input Folder)
- **ImageExt** Input image extension
- **Mm** Input image month
- **Time** Input image time
- **Yy** Input image year (2 digits)
- **Yyyy** Input image year (4 digits)
- **Barcode** The barcode value.

For example,

- `<Barcode>.<Ext>` The barcode value is used as the name of the output file.
- `<Barcode>/<FileName>.<Ext>` The barcode value is used as the name of the folder. This is useful for sorting documents based on barcode values.
- `<Filename>.<ImageExt>.<Ext>` The input image extension is preserved in the output file name.
- `OCR_<FileName>.<Ext>`
The prefix "OCR_" is added to the output file name.

The default value is <FileName>.<Ext>.

**KeepLastModifiedDate**

- **Type**: Boolean
  - Specifies whether the date when the image file was modified must be retained when exporting the data.
  - The default value is FALSE.

**OutputLocation**

- **Type**: String
  - Specifies the path to the output location (a shared folder, e-mail addresses, or a folder in the SharePoint document library, according to the value of the PublishingMethod property).
  - Multiple e-mail addresses must be separated by commas.

**PublishingMethod**

- **Type**: OutputPublishingMethodEnum
  - Specifies the destination. The default value is OPM_SharedFolder.

**SharePointServerUrl**

- **Type**: String
  - Specifies URL address of the SharePoint server site. This property is used only if the value of the PublishingMethod property is set to OPM_SharePoint.

**SharePointDocumentLibrary**

- **Type**: String
  - Specifies the SharePoint document library. This property is used only if the value of the PublishingMethod property is set to OPM_SharePoint.

**SMTPSettings**

- **Type**: SMTPSettings
  - Specifies the parameters for connection with the SMTP e-mail server. This property is used only if the value of the PublishingMethod property is set to OPM_Smtp.

**Note**: ABBYY Recognition Server only supports SMTP servers that use the PLAIN authentication method.

**Output parameter**

The **OutputFormatSettings** object is the output parameter of the IClient::CreateOutputFormatSettings method.
See also

ExportParams
ABBYY Recognition Server 4 COM-based API
RTFExportSettings Object
(IRTFExportSettings Interface)

This object provides functionality for tuning export parameters when exporting recognized text to RTF format. The IRTFExportSettings interface is a child object of the IOutputFormatSettings interface and inherits all its properties. It is an element of the OutputFormatSettingsCollection object.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ForceFixedPageSize</td>
<td>Boolean</td>
<td>Specifies whether export result must fit the dimensions set by the PaperWidth and PaperHeight properties. The default value is FALSE.</td>
</tr>
<tr>
<td>HighlightErrorsWithBackgroundColor</td>
<td>Boolean</td>
<td>Specifies if uncertainly recognized symbols are highlighted with background color when exported in RTF format. The default value is FALSE.</td>
</tr>
<tr>
<td>PaperHeight</td>
<td>Long</td>
<td>Stores paper height in twips (1/1440 of inch). The default value is the height of A4 format page. See the table below.</td>
</tr>
<tr>
<td>PaperWidth</td>
<td>Long</td>
<td>Stores paper width in twips (1/1440 of inch). The default value is the width of A4 format page. See the table below.</td>
</tr>
<tr>
<td>RTFSynthesisMode</td>
<td>RTFSynthesisModeEnum</td>
<td>Specifies the mode of RTF file synthesis from the recognized text when exporting to RTF formats. The default value is RSM_RTFColumns.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Specifies whether pictures are to be written in files in RTF format. The default value is TRUE. Note: The format in which pictures are saved in the output file is selected.</td>
</tr>
</tbody>
</table>
WritePictures: Boolean

Automatically basing on the two picture properties: Color Type (black and white, grayscale or color) and Color Variety (low or high). Black and white pictures are always saved in PNG format. Grayscale and color pictures are saved in PNG format in the case of low color variety, and in JPEG format in the case of high color variety.

### Paper size in different units of measurement

<table>
<thead>
<tr>
<th>Paper size</th>
<th>in inch</th>
<th>in mm</th>
<th>in twips (1/1440 of inch)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A3</td>
<td>11,69 x 16,54</td>
<td>297 x 420</td>
<td>16838 x 23811</td>
</tr>
<tr>
<td>A4</td>
<td>8,27 x 11,69</td>
<td>210 x 297</td>
<td>11909 x 16834</td>
</tr>
<tr>
<td>A5</td>
<td>5,83 x 8,27</td>
<td>148 x 210</td>
<td>8391 x 11909</td>
</tr>
<tr>
<td>Legal</td>
<td>8,5 x 14</td>
<td>216 x 356</td>
<td>12240 x 20160</td>
</tr>
<tr>
<td>Letter</td>
<td>8,5 x 11</td>
<td>216 x 279</td>
<td>12240 x 15840</td>
</tr>
<tr>
<td>Executive</td>
<td>7,25 x 10,5</td>
<td>184 x 266</td>
<td>10440 x 15120</td>
</tr>
</tbody>
</table>

See also

- `OutputFormatSettings`
- `OutputFormatSettingsCollection`
ABBYY Recognition Server 4 COM-based API
MSWordExportSettings Object (IMSWordExportSettings Interface)

This object provides functionality for tuning export parameters when exporting recognized text to DOC format. The IMSWordExportSettings interface is a child object of the IOutputFormatSettings interface and inherits all its properties. It is an element of the OutputFormatSettingsCollection collection.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ForceFixedPageSize</td>
<td>Boolean</td>
<td>Specifies whether export result must fit the dimensions set by the PaperWidth and PaperHeight properties. The default value is FALSE.</td>
</tr>
<tr>
<td>HighlightErrorsWithBackgroundColor</td>
<td>Boolean</td>
<td>Specifies if uncertainly recognized symbols are highlighted with background color when exported in DOC format. The default value is FALSE.</td>
</tr>
<tr>
<td>PaperHeight</td>
<td>Long</td>
<td>Stores paper height in twips (1/1440 of inch). The default value is the height of A4 format page. See the table &quot;Paper size in different units of measurement&quot;.</td>
</tr>
<tr>
<td>PaperWidth</td>
<td>Long</td>
<td>Stores paper width in twips (1/1440 of inch). The default value is the width of A4 format page. See the table &quot;Paper size in different units of measurement&quot;.</td>
</tr>
<tr>
<td>RTFSynthesisMode</td>
<td>RTFSynthesisModeEnum</td>
<td>Specifies the mode of DOC file synthesis from the recognized text when exporting to DOC formats. The default value is RSM_RTFColumns. Specifies whether pictures are to be written in files in DOC formats.</td>
</tr>
</tbody>
</table>
WritePictures: Boolean

*Note:* The format in which pictures are saved in the output file is selected automatically basing on the two picture properties: Color Type (black and white, grayscale or color) and Color Variety (low or high). Black and white pictures are always saved in PNG format. Grayscale and color pictures are saved in PNG format in the case of low color variety, and in JPEG format in the case of high color variety.

See also

*OutputFormatSettings*, *OutputFormatSettingsCollection*
**DOCXExportSettings Object**

(IDOCXExportSettings Interface)

This object provides functionality for tuning export parameters when exporting recognized text to DOCX format. The IDOCXExportSettings interface is a child object of the IOutputFormatSettings interface and inherits all its properties. It is an element of the OutputFormatSettingsCollection collection.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ForceFixedPageSize</td>
<td>Boolean</td>
<td>Specifies whether export result must fit the dimensions set by the PaperWidth and PaperHeight properties. The default value is FALSE.</td>
</tr>
<tr>
<td>HighlightErrorsWithBackgroundColor</td>
<td>Boolean</td>
<td>Specifies if uncertainly recognized symbols are highlighted with background color when exported in DOCX format. The default value is FALSE.</td>
</tr>
<tr>
<td>PaperHeight</td>
<td>Long</td>
<td>Stores paper height in twips (1/1440 of inch). The default value is the height of A4 format page. See the table &quot;Paper size in different units of measurement&quot;.</td>
</tr>
<tr>
<td>PaperWidth</td>
<td>Long</td>
<td>Stores paper width in twips (1/1440 of inch). The default value is the width of A4 format page. See the table &quot;Paper size in different units of measurement&quot;.</td>
</tr>
<tr>
<td>RTFSynthesisMode</td>
<td>RTFSynthesisModeEnum</td>
<td>Specifies the mode of DOCX file synthesis from the recognized text when exporting to DOCX formats. The default value is RSM_RTFColumns. Specifies whether pictures are to be written in files in DOCX</td>
</tr>
</tbody>
</table>
WritePictures  Boolean

format. The default value is TRUE. Note: The format in which pictures are saved in the output file is selected automatically basing on the two picture properties: Color Type (black and white, grayscale or color) and Color Variety (low or high). Black and white pictures are always saved in PNG format. Grayscale and color pictures are saved in PNG format in the case of low color variety, and in JPEG format in the case of high color variety.

See also

OutputFormatSettings, OutputFormatSettingsCollection
ABBYY Recognition Server 4 COM-based API
XLExportSettings Object (IXLExportSettings Interface)

This object provides functionality for tuning export parameters when exporting recognized text to XLS format. The IXLExportSettings interface is a child object of the IOutputFormatSettings interface and inherits all its properties. It is an element of the OutputFormatSettingsCollection collection.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConvertNumericValuesToNumbers</td>
<td>Boolean</td>
<td>Specifies whether numerical values in recognized text are to be exported to XLS format as numbers. The default value is TRUE.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>If this property is set to TRUE, only text from table blocks is exported into XLS format. The default value is FALSE.</td>
</tr>
</tbody>
</table>

See also

OutputFormatSettings, OutputFormatSettingsCollection
ABBYY Recognition Server 4 COM-based API
**XLSXExportSettings Object**

(IXLSXExportSettings Interface)

This object provides functionality for tuning export parameters when exporting recognized text to XLSX format. The **IXLSXExportSettings** interface is a child object of the **IOutputFormatSettings** interface and inherits all its properties. It is an element of the **OutputFormatSettingsCollection** collection.

### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConvertNumericValuesToNumbers</td>
<td>Boolean</td>
<td>Specifies whether numerical values in recognized text are to be exported to XLSX format as numbers. The default value is TRUE.</td>
</tr>
<tr>
<td>IgnoreTextOutsideTables</td>
<td>Boolean</td>
<td>If this property is set to TRUE, only text from table blocks is exported into XLSX format. The default value is FALSE.</td>
</tr>
</tbody>
</table>

### See also

- **OutputFormatSettings**
- **OutputFormatSettingsCollection**
ABBYY Recognition Server 4 COM-based API
**HTMLExportSettings Object**

(IHTMLExportSettings Interface)

This object provides functionality for tuning export parameters when exporting recognized text to HTML format. The IHTMLExportSettings interface is a child object of the IOutputFormatSettings interface and inherits all its properties. It is an element of the OutputFormatSettingsCollection collection.

### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AllowCss</td>
<td>Boolean</td>
<td>Specifies if a built-in style sheet (CSS) is to be used. CSS requires Internet Explorer 4.0 or later.</td>
</tr>
<tr>
<td>CodePage</td>
<td>CodePageEnum</td>
<td>This property sets the code page to which the recognized text is to be exported. If this property does not specify any code page (CP_Null), the code page is selected automatically. The default value is CP_Null.</td>
</tr>
<tr>
<td>EncodingType</td>
<td>TextEncodingTypeEnum</td>
<td>Specifies the encoding type of the output file in HTML format. This property is TET_Simple by default which means that each symbol is coded by one byte in the output file. Note: If this property is set to a value different from TET_Simple, the CodePage property is ignored during export.</td>
</tr>
<tr>
<td>HTMLSynthesisMode</td>
<td>HTMLSynthesisModeEnum</td>
<td>Specifies a mode of synthesizing HTML code from the recognized text. There exist three modes of synthesis: retain paragraphs only, retain paragraphs and fonts, retain full page layout. The default value is HSM_PageLayout, which means that the entire layout is retained.</td>
</tr>
<tr>
<td>WritePictures</td>
<td>Boolean</td>
<td>Specifies whether pictures must be saved along with the file in HTML format. If pictures are not written, references to them in HTML files are also omitted. The default value is TRUE. Note: The format in which pictures are saved in the output file is selected automatically basing on the two picture properties: Color Type (black and white, grayscale or color) and</td>
</tr>
</tbody>
</table>
Color Variety (low or high). Black and white pictures are always saved in PNG format. Grayscale and color pictures are saved in PNG format in the case of low color variety, and in JPEG format in the case of high color variety.

See also

OutputFormatSettings, OutputFormatSettingsCollection
XMLExportSettings Object
(IXMLExportSettings Interface)

This object provides functionality for tuning export parameters when exporting recognized text to XML format. The **IXMLExportSettings** interface is a child object of the **IOutputFormatSettings** interface and inherits all its properties. It is an element of the **OutputFormatSettingsCollection** collection. You can see an XML scheme for an XML Document in the ExportToXml.xsd file which can be found in the Bin subfolder of the ABBYY Recognition Server folder. Also, see the Document XML Scheme and XML scheme representation for details.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PagesPerFile</td>
<td>Long</td>
<td>Specifies the maximum number of pages to be included in one file.</td>
</tr>
<tr>
<td>WriteCharactersFormatting</td>
<td>Boolean</td>
<td>Specifies whether character formatting (e.g. font size, font style) is to be written to files in XML format. The default value is FALSE.</td>
</tr>
<tr>
<td>WriteCharAttributes</td>
<td>Boolean</td>
<td>Specifies whether character attributes (e.g. character coordinates) are to be written to files in XML format. The default value is FALSE.</td>
</tr>
<tr>
<td>WriteExtendedCharAttributes</td>
<td>Boolean</td>
<td>Specifies whether extended attributes (e.g. whether a character was recognized uncertainly, whether the word was found in the dictionary) are to be written to files in XML format. The default value is FALSE.</td>
</tr>
<tr>
<td>WriteNonDeskewedCoordinates</td>
<td>Boolean</td>
<td>Specifies whether character coordinates written to files in XML format are coordinates on a non-deskewed image plane. The default value is FALSE.</td>
</tr>
</tbody>
</table>

See also

[OutputFormatSettings](#), [OutputFormatSettingsCollection](#)
ABBYY Recognition Server 4 COM-based API
TextExportSettings Object
(ITextExportSettings Interface)

This object provides functionality for tuning export parameters when exporting recognized text to TXT format. The ITextExportSettings interface is a child object of the IOutputFormatSettings interface and inherits all its properties. It is an element of the OutputFormatSettingsCollection collection.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CodePage</td>
<td>CodePageEnum</td>
<td>This property specifies the code page to which the recognized text is to be exported. The default value is CP_Null.</td>
</tr>
<tr>
<td>EncodingType</td>
<td>TextEncodingTypeEnum</td>
<td>This property specifies the encoding type of the output file in TXT format. The default value is TET_Simple. Note: If this property is set to a value different from TET_Simple, the CodePage property is ignored during export.</td>
</tr>
<tr>
<td>ExportParagraphsAsOneLine</td>
<td>Boolean</td>
<td>Specifies whether each paragraph in the recognized text is to be exported as one line. The default value is FALSE.</td>
</tr>
<tr>
<td>InsertEmptyLineBetweenParagraphs</td>
<td>Boolean</td>
<td>Specifies if an empty line should be inserted between paragraphs. The default value is FALSE.</td>
</tr>
<tr>
<td>KeepOriginalHeadersFooters</td>
<td>Boolean</td>
<td>If this property is set to TRUE, original headers and footers will be preserved in the output file. The default value is TRUE.</td>
</tr>
<tr>
<td>UsePageBreaks</td>
<td>Boolean</td>
<td>Specifies whether page break symbols (0x12) must be inserted between pages when multiple pages are exported into TXT format. The default value is FALSE.</td>
</tr>
</tbody>
</table>
See also

OutputFormatSettings, OutputFormatSettingsCollection
ABBYY Recognition Server 4 COM-based API
CSVExportSettings Object (ICSVExportSettings Interface)

This object provides functionality for tuning export parameters when exporting recognized text to CSV format. The ICSVExportSettings interface is a child object of the IOutputFormatSettings interface and inherits all its properties. It is an element of the OutputFormatSettingsCollection collection.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CodePage</td>
<td>CodePageEnum</td>
<td>This property specifies the code page to which the recognized text is to be exported. The default value is CP_Null.</td>
</tr>
<tr>
<td>EncodingType</td>
<td>TextEncodingTypeEnum</td>
<td>This property specifies the encoding type of the output file in CSV format. The default value is TET_Simple. Note: If this property is set to a value different from TET_Simple, the CodePage property is ignored during export.</td>
</tr>
<tr>
<td>IgnoreTextOutsideTables</td>
<td>Boolean</td>
<td>If this property is set to TRUE, only text from table blocks is exported into CSV format. The default value is FALSE.</td>
</tr>
<tr>
<td>TabSeparator</td>
<td>String</td>
<td>Stores the character with which the table separators are to be replaced in the exported text. The string accessed through this property must contain only one character from the ASCII character set. The default value is &quot;TAB&quot; character ('\t' in C/C++ or vbTab in Visual Basic).</td>
</tr>
<tr>
<td>UsePageBreaks</td>
<td>Boolean</td>
<td>Specifies if page break symbols (0x12) must be inserted between pages if pages are to be exported into CSV format. The default value is FALSE.</td>
</tr>
</tbody>
</table>

See also

OutputFormatSettings, OutputFormatSettingsCollection
PDFExportSettings Object (IPDFExportSettings Interface)

This object provides functionality for tuning export parameters when exporting recognized text to PDF format. The IPDFExportSettings interface is a child object of the IOutputFormatSettings interface and inherits all its properties. It is an element of the OutputFormatSettingsCollection collection.

When you save texts that use a non-Latin codepage (say, Cyrillic, Greek, Czech, etc.), ABBYY Recognition Server Open API uses the fonts provided by ParaType company (http://www.paratype.com/store/).

Peculiarities of exporting hieroglyphic languages to PDF

You can export hieroglyphic languages to PDF in any mode other than PDF Image Only (ExportMode = PEM_ImageOnly). For correct depicting of hieroglyphic characters, files for East Asian languages must be installed on all computers where the Processing Station component is installed.

When exporting to PDF, the following fonts are used:

- for the ChineseTraditional recognition language – the PMingLiU font and the MingLiU monospace font;
- for the ChineseSimplified recognition language – the SimSun font;
- for the Japanese recognition language – the MS PMincho font and the MS Mincho monospace font;
- for the Korean recognition language – the Batang font and the BatangChe monospace font.

Note: The earliest version of the PDF file which matches the specified properties of the PDFEncryptionInfo object is selected as the version of the PDF file.

- The earliest file version available is version 1.3.
- If at least one of the AllowFillingFormFields, AllowExtractingTextAndGraphicsExt, AllowDocumentAssembling or AllowPrintingExt properties is TRUE, or the EncryptionLevel property is PEL_High, the PDF file version will be 1.4.
- If the EncryptionLevel property is PEL_HighAES, the version will be 1.6.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AuthorName</td>
<td>String</td>
<td>Stores the name of the author of the PDF document. You may set this...</td>
</tr>
</tbody>
</table>
### Author
**String**

Sets a color mode for output PDF file: Auto, ColorToBwBinary, ColorToGray or SaveColoration. **Note:** These settings will be ignored if UseImprovedCompression is TRUE.

### ColorControlType
**PictureColorControlTypeEnum**

Sets a color mode for output PDF file: Auto, ColorToBwBinary, ColorToGray or SaveColoration.

### DocumentInfo
**DocumentInfoItems**, read-only

Stores the metadata from the PDF file.*

### EncryptionInfo
**PDFEncryptionInfo**, read-only

Specifies encryption parameters of the PDF file. The property returns a reference to the PDFEncryptionInfo object.

### HeaderAndFooter
**HeaderAndFooterSettings**

Specifies the parameters of the header and footer for the PDF document.

### IsEncryptionRequested
**Boolean**

Indicates whether the PDF file must be encrypted. The default value is FALSE. If this property is set to TRUE, original headers and footers will be preserved in the output file. The default value is TRUE.

### KeepOriginalHeadersFooters
**Boolean**

Stores the keywords of the PDF document.*. The default value is an empty string.

### Keywords
**String**

Specifies if document metadata (the key-value pairs of the DocumentInfo property and the values of the Title, Author, Subject, Producer, Keywords properties*) must be used instead of the properties of the source document. The
<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PaperHeight</td>
<td>Long</td>
<td>Stores paper height in twips (1/1440 of inch). The default value is the height of A4 format page. See the table &quot;Paper size in different units of measurement&quot;. The value of this property is ignored, if the UseOriginalPaperSize property is set to TRUE.</td>
</tr>
<tr>
<td>PaperWidth</td>
<td>Long</td>
<td>Stores paper width in twips (1/1440 of inch). The default value is the width of A4 format page. See the table &quot;Paper size in different units of measurement&quot;. The value of this property is ignored, if the UseOriginalPaperSize property is set to TRUE.</td>
</tr>
<tr>
<td>PDFExportMode</td>
<td>PDFExportModeEnum</td>
<td>Specifies the mode of export of recognized text into PDF format. This may be: text and pictures only, text over the page image, text under the page image, page image only. The default value is PEM_ImageOnText.</td>
</tr>
<tr>
<td>PDFVersion</td>
<td>PDFVersionEnum</td>
<td>Specifies a PDF version.</td>
</tr>
<tr>
<td>PictureResolution</td>
<td>Long</td>
<td>Stores the value of picture resolution in dpi, which is used for exporting pictures into PDF format. This property may be set to -1, which means that the default value is FALSE. Note: If at least one of the Title, Author, Subject, Producer, Keywords, and DocumentInfo properties is changed, this property is automatically set to TRUE.</td>
</tr>
</tbody>
</table>
original resolution must be preserved. The default value is 96 dpi.

Stores the name of the producer of the PDF document. The default value is "ABBYY Recognition Server".

Stores the value in percentage points of the JPEG quality for color pictures saved in PDF format. This value is ignored for black-and-white pictures. The default value is 50%.

Specifies if uncertainly recognized words will be replaced with their images when exporting into PDF format. You may use this property when the ExportMode property is set to PEM_TextAndPictures or PEM_TextOnImage, otherwise its value is ignored. The default value is FALSE.

Specifies the export profile. This may be: BestQuality, MinSize, MaxSpeed or Balanced. The default value is Balanced.

Stores the subject of the PDF document*. The default value is an empty string.

Stores the title of the PDF document*. The default value is an empty string.

If this property is TRUE, the images in the PDF file will be compressed using

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Producer</td>
<td>String</td>
<td>Stores the name of the producer of the PDF document. The default value is &quot;ABBYY Recognition Server&quot;.</td>
</tr>
<tr>
<td>Quality</td>
<td>Long</td>
<td>Stores the value in percentage points of the JPEG quality for color pictures saved in PDF format. This value is ignored for black-and-white pictures. The default value is 50%.</td>
</tr>
<tr>
<td>ReplaceUncertainWordsWithImages</td>
<td>Boolean</td>
<td>Specifies if uncertainly recognized words will be replaced with their images when exporting into PDF format. You may use this property when the ExportMode property is set to PEM_TextAndPictures or PEM_TextOnImage, otherwise its value is ignored. The default value is FALSE.</td>
</tr>
<tr>
<td>Scenario</td>
<td>String</td>
<td>Specifies the export profile. This may be: BestQuality, MinSize, MaxSpeed or Balanced. The default value is Balanced.</td>
</tr>
<tr>
<td>Subject</td>
<td>String</td>
<td>Stores the subject of the PDF document*. The default value is an empty string.</td>
</tr>
<tr>
<td>Title</td>
<td>String</td>
<td>Stores the title of the PDF document*. The default value is an empty string.</td>
</tr>
</tbody>
</table>
UseImprovedCompression  Boolean

Mixed Raster Content (MRC) technology. MRC will provide better compression for some images than the commonly used JPEG method. For example, MRC will better compress good quality images with high-contrast text against a uniform background. Use MRC to reduce the size of the resulting PDF file. The default value is FALSE.

Note: When using MRC on low contrast images with a parti-coloured background artifacts are possible.

If this property is TRUE, original paper size is retained during export to PDF format. If the value of this property is FALSE, the paper size specified in the PaperWidth and PaperHeight properties is used. The default value is TRUE.

UseOriginalPaperSize  Boolean

WriteAnnotations (Obsolete)  Boolean

Specifies whether to write annotations when creating a PDF file. The default value is TRUE.

WriteTaggedPdf  Boolean

Specifies if the recognized text should be exported to tagged PDF. Tagged PDF is a particular use of structured PDF that allows page content to be extracted and used for various purposes such as reflow of text and graphics, conversion to file formats such as HTML and XML, and accessibility to the
* – **Note:** If you change the values of the **Author**, **Keywords**, **Subject** or **Title** properties the values of the corresponding metadata keys of the **DocumentInfo** property will be changed or the corresponding key-value pairs will be added to the collection. Vice versa, if the values of the Author, Keywords, Subject or Title keys of the **DocumentInfo** property are set to some value, the values of the corresponding properties of the **PDFExportSettings** object will be set to the same value.

**See also**

- [OutputFormatSettings](#)
- [OutputFormatSettingsCollection](#)
ABYY Recognition Server 4 COM-based API
PDFAEExportSettings Object
(IPDFAEExportSettings Interface)

This object provides functionality for tuning export parameters when exporting recognized text to PDF/A format. The **IPDFAEExportSettings** interface is a child object of the **IOutputFormatSettings** interface and inherits all its properties. It is an element of the **OutputFormatSettingsCollection** collection.

When you save texts that use a non-Latin codepage (say, Cyrillic, Greek, Czech, etc.), ABBYY Recognition Server Open API uses the fonts provided by ParaType company ([http://www.paratype.com/store/](http://www.paratype.com/store/)).

**Peculiarities of exporting hieroglyphic languages to PDF/A**

You can export hieroglyphic languages to PDF/A in any mode other than PDF Image Only (PDFExportMode = PEM_ImageOnly). For correct depicting of hieroglyphic characters, files for East Asian languages must be installed on all computers where the Processing Station component is installed.

When exporting to PDF/A, the following fonts are used:

- for the ChineseTraditional recognition language – the PMingLiU font and the MingLiU monospace font;
- for the ChineseSimplified recognition language – the SimSun font;
- for the Japanese recognition language – the MS PMincho font and the MS Mincho monospace font;
- for the Korean recognition language – the Batang font and the BatangChe monospace font.

**Properties**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Author</td>
<td><strong>String</strong></td>
<td>Stores the name of the author of the PDF/A document*. You may set this property to the name of the user. The default value is an empty string.</td>
</tr>
<tr>
<td>ColorControlType</td>
<td><strong>PictureColorControlTypeEnum</strong></td>
<td>Sets a color mode for output PDF file: Auto, ColorToBwBinary, ColorToGray or SaveColoration. <strong>Note:</strong> These settings will be ignored if <strong>UseImprovedCompression</strong> is set.</td>
</tr>
</tbody>
</table>
DocumentInfo

**DocumentInfoItems**, read-only

Stores the metadata from the PDF/A file.*

**HeaderAndFooter**

**HeaderAndFooterSettings**

Specifies the parameters of the header and footer for the PDF/A document.

If this property is set to TRUE, original headers and footers will be preserved in the output file. The default value is TRUE.

KeepOriginalHeadersFooters **Boolean**

Stores the keywords of the PDF/A document*. The default value is an empty string.

Specifies if document metadata (the key-value pairs of the **DocumentInfo** property and the values of the **Title**, **Author**, **Subject**, **Producer**, **Keywords** properties*) must be used instead of the properties of the source document. The default value is FALSE.

Note: If at least one of the **Title**, **Author**, **Subject**, **Producer**, **Keywords**, and **DocumentInfo** properties is changed, this property is automatically set to TRUE.

**OverwriteMetadata** **Boolean**

Stores paper height in twips (1/1440 of inch). The default value is the height of A4 format page.

See the table "Paper size in different units of measurement". The value of this property is ignored, if the **UseOriginalPageSize** property is set to TRUE.

**PaperHeight** **Long**

Stores paper width in twips (1/1440 of inch). The default value is the width of A4 format page. See the table "Paper size in different units of measurement". The value of this property is ignored, if the **UseOriginalPageSize** property is set to TRUE.

**PaperWidth** **Long**
Specifies the mode of export of recognized text into PDF/A format. It may be: text under the page image (PEM_ImageOnText) and page image only (PEM_ImageOnly). The default value is PEM_ImageOnText.

Specifies the PDF/A standard to be used for the output file.

Specifies a PDF version.

Stores the value of picture resolution in dpi, which is used for exporting pictures into PDF/A format. This property may be set to -1, which means that the original resolution must be preserved. The default value is 96 dpi.

Stores the name of the producer of the PDF/A document. The default value is "ABBYY Recognition Server".

Stores the value in percentage points of the JPEG quality for color pictures saved in PDF/A format. This value is ignored for black-and-white pictures. The default value is 50%.

Specifies the export profile. This may be: BestQuality, MinSize, MaxSpeed or Balanced. The default value is Balanced.

Stores the subject of the PDF/A document. The default value is an empty string*.

Stores the title of the PDF/A document. The default value is an empty string*.

If this property is TRUE, the images in the PDF file will be compressed using Mixed Raster Content (MRC) technology. MRC will provide better compression.
UseImprovedCompression  

**Boolean**

for some images than the commonly used JPEG method. For example, MRC will better compress good quality images with high-contrast text against a uniform background. Use MRC to reduce the size of the resulting PDF file. The default value is FALSE.

**Note:** When using MRC on low contrast images with a parti-coloured background artifacts are possible.

If this property is TRUE, original paper size is retained during export to PDF/A format. If the value of this property is FALSE, the paper size specified in the **PaperWidth** and **PaperHeight** properties is used. The default value is TRUE.

WriteAnnotations (Obsolete)  

**Boolean**

Specifies whether to write annotations when creating a PDF file. The default value is TRUE.

*—  Note: If you change the values of the Author, Keywords, Subject or Title properties the values of the corresponding metadata keys of the DocumentInfo property will be changed or the corresponding key-value pairs will be added to the collection. Vice versa, if the values of the Author, Keywords, Subject or Title keys of the DocumentInfo property are set to some value, the values of the corresponding properties of the PDFAExportSettings object will be set to the same value.

See also

**OutputFormatSettings**,  
**OutputFormatSettingsCollection**
EPUBExportSettings Object (IPDFExportSettings Interface)

This object provides functionality for tuning export parameters when exporting recognized text to EPUB format. The IEPUBExportSettings interface is a child object of the IOutputFormatSettings interface and inherits all its properties. It is an element of the OutputFormatSettingsCollection collection.

When you save texts that use a non-Latin codepage (say, Cyrillic, Greek, Czech, etc.), ABBYY Recognition Server Open API uses the fonts provided by ParaType company (http://www.paratype.com/store/).

Peculiarities of exporting hieroglyphic languages to EPUB

You can export hieroglyphic languages to EPUB. For correct depicting of hieroglyphic characters, files for East Asian languages must be installed on all computers where the Processing Station component is installed.

When exporting to EPUB, the following fonts are used:

- for the ChineseTraditional recognition language – the PMingLiU font and the MingLiU monospace font;
- for the ChineseSimplified recognition language – the SimSun font;
- for the Japanese recognition language – the MS PMincho font and the MS Mincho monospace font;
- for the Korean recognition language – the Batang font and the BatangChe monospace font.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EpubVersion</td>
<td>EpubVersionEnum</td>
<td>Specifies a EPUB version.</td>
</tr>
<tr>
<td>FontFormattingMode</td>
<td>FontFormattingModeEnum</td>
<td>Specifies a layout retention method for files saved in EPUB format. The default value of this property is FAM_Restricted.</td>
</tr>
<tr>
<td>KeepPictures</td>
<td>Boolean</td>
<td>Specifies whether to retain pictures in the EPUB file. The default value is TRUE.</td>
</tr>
<tr>
<td>PictureFormat</td>
<td>ExportPictureFormatEnum</td>
<td>Specifies the color mode and compression method for images inside the EPUB file. The default value is EPF_Automatic. Stores the value of picture resolution in dpi, which is used for exporting pictures into</td>
</tr>
</tbody>
</table>
PictureResolution **Long**

EPUB format. This property may be set to -1, which means that the original resolution must be preserved. The default value is 96 dpi.

Stores the value in percentage points of the JPEG quality for color pictures saved in EPUB format. This value is ignored for black-and-white pictures. The default value is 50%.

Specifies whether the first page should be treated as a cover. The default value is TRUE.

See also

**OutputFormatSettings**, **OutputFormatSettingsCollection**
ABBYY Recognition Server 4 COM-based API
This object provides functionality for tuning export parameters when exporting to TIFF format. The `ITiffExportSettings` interface is a child object of the `IOutputFormatSettings` interface and inherits all its properties. It is an element of the `OutputFormatSettingsCollection` collection.

### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ColorMode</td>
<td><code>ImageColorModeEnum</code></td>
<td>Specifies image color. The default value is ICM_AsIs.</td>
</tr>
<tr>
<td>Compression</td>
<td><code>ImageCompressionTypeEnum</code></td>
<td>Specifies the compression. If the ColorMode property is set to ICM_AsIs, ICM_Color, or ICM_Gray, this property can have one of the following values: ICT_Uncompressed, ICT_Zip, ICT_Jpeg. If the ColorMode property is set to ICM_BlackAndWhite, this property can have one of the following values: ICT_Uncompressed, ICT_Zip, ICT_Group3, ICT_Group4, ICT_PackBits. The default value is ICT_Zip.</td>
</tr>
<tr>
<td>Resolution</td>
<td><code>Long</code></td>
<td>Specifies the value of picture resolution in dpi, which is used for exporting. The default value is -1 which means that the original resolution must be preserved.</td>
</tr>
</tbody>
</table>

See also

`OutputFormatSettings`, `OutputFormatSettingsCollection`
Jpeg2kExportSettings Object (IJpeg2kExportSettings Interface)

This object provides functionality for tuning export parameters when exporting to JPEG 2000 format. The IJpeg2kExportSettings interface is a child object of the IOutputFormatSettings interface and inherits all its properties. It is an element of the OutputFormatSettingsCollection collection.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ColorMode</td>
<td>ImageColorModeEnum</td>
<td>Specifies image color. This property can be set to one of the following values: ICM_AsIs, ICM_Color, ICM_Gray. The default value is ICM_AsIs.</td>
</tr>
<tr>
<td>Quality</td>
<td>Long</td>
<td>Specifies the quality in percentage points. The default value is 80.</td>
</tr>
<tr>
<td>Resolution</td>
<td>Long</td>
<td>Stores the value of picture resolution in dpi, which is used for exporting. The default value is -1 which means that the original resolution must be preserved.</td>
</tr>
</tbody>
</table>

See also

OutputFormatSettings, OutputFormatSettingsCollection
ABBYY Recognition Server 4 COM-based API
JpegExportSettings Object
(IJpegExportSettings Interface)

This object provides functionality for tuning export parameters when exporting to JPEG format. The IJpegExportSettings interface is a child object of the IOutputFormatSettings interface and inherits all its properties. It is an element of the OutputFormatSettingsCollection collection.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ColorMode</td>
<td>ImageColorModeEnum</td>
<td>Specifies image color. This property can be set to one of the following values: ICM_AsIs, ICM_Color, ICM_Gray. The default value is ICM_AsIs.</td>
</tr>
<tr>
<td>Quality</td>
<td>Long</td>
<td>Specifies the quality in percentage points. The default value is 80.</td>
</tr>
<tr>
<td>Resolution</td>
<td>Long</td>
<td>Stores the value of picture resolution in dpi, which is used for exporting. The default value is -1 which means that the original resolution must be preserved.</td>
</tr>
</tbody>
</table>

See also

OutputFormatSettings, OutputFormatSettingsCollection
ABBYY Recognition Server 4 COM-based API
JBig2ExportSettings Object (IJBig2ExportSettings Interface)

This object provides functionality for tuning export parameters when exporting to JBIG2 format. The IJBig2ExportSettings interface is a child object of the IOutputFormatSettings interface and inherits all its properties. It is an element of the OutputFormatSettingsCollection collection.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resolution</td>
<td>Long</td>
<td>Specifies the value of picture resolution in dpi, which is used for exporting. The default value is -1 which means that the original resolution must be preserved.</td>
</tr>
</tbody>
</table>

See also

OutputFormatSettings, OutputFormatSettingsCollection
ABBYY Recognition Server 4 COM-based API
AltoExportSettings Object
(IAltoExportSettings Interface)

This object provides functionality for tuning export parameters when exporting recognized text to Alto XML format. The IAltoExportSettings interface is a child object of the IOutputFormatSettings interface and inherits all its properties. It is an element of the OutputFormatSettingsCollection collection.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TextCoordinatesParticularity</td>
<td>TextCoordinatesParticularityEnum</td>
<td>Specifies how text should be divided: by words or by lines. This property can be set to one of the following values: TCP_Words, TCP_Lines. The default value is TCP_Words.</td>
</tr>
<tr>
<td>FontFormattingMode</td>
<td>FontFormattingModeEnum</td>
<td>Select the desired font formatting mode: plain, restricted or full. This property can be set to one of the following values: FFM_Full, FFM_Plain, FFM_Restricted. The default value is FFM_Plain.</td>
</tr>
<tr>
<td>MeasurementUnit</td>
<td>AltoMeasurementUnitEnum</td>
<td>Specifies the measurement unit used to describe size and coordinates of objects in the output XML file. This property can be set to one of the following values: AMU_Inch1200, AMU_Mm10, AMU_Pixel. The default value is AMU_Pixel.</td>
</tr>
</tbody>
</table>

See also

OutputFormatSettings, OutputFormatSettingsCollection
ABBYY Recognition Server 4 COM-based API
StringsCollection Object (IStringsCollection Interface)

This object provides access to a collection of strings.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count</td>
<td>Long</td>
<td>read-only Stores the number of elements in the collection.</td>
</tr>
</tbody>
</table>

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add</td>
<td>Adds a new element at the end of the collection.</td>
</tr>
<tr>
<td>Item</td>
<td>Provides access to a single element of the collection.</td>
</tr>
<tr>
<td>Remove</td>
<td>Removes an element from the collection.</td>
</tr>
<tr>
<td>RemoveAll</td>
<td>Removes all the elements from the collection.</td>
</tr>
</tbody>
</table>

See also

IClient::Workflows,
IInputFile::Errors,
IInputFile::Warnings,
IXmlResult::Errors,
IXmlResult::Warnings,
IOutputDocument::FileNames,
IRecognitionParams::Languages,
Working with Collections.

See samples: Hello, Listening, AsyncProcessing, Sample for ASP.NET.
ABBYY Recognition Server 4 COM-based API
Statistics Object (IStatistics Interface)

This object represents statistics about the results of recognition.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PagesArea</td>
<td>Long, read-only</td>
<td>Returns the total area of recognized pages measured in A4 pages. This is the number that was used to increment the page counter for the license.</td>
</tr>
<tr>
<td>TotalCharacters</td>
<td>Long, read-only</td>
<td>Returns the total number of recognized characters.</td>
</tr>
<tr>
<td>UncertainCharacters</td>
<td>Long, read-only</td>
<td>Returns the total number of uncertain characters.</td>
</tr>
</tbody>
</table>

See also

IXmlResult::Statistics
ABYY Recognition Server 4 COM-based API
PDFEncryptionInfo Object (IPDFEncryptionInfo Interface)

This object provides access to encryption parameters of the PDF file during export. These parameters are set in the EncryptionInfo property of PDFExportSettings. The PDFEncryptionInfo object allows you to do the following:

- set owner and user passwords;
- set the level of encryption;
- enable or disable the following:
  - adding or modifying text annotations and interactive form fields;
  - assembling the document: inserting, rotating, or deleting pages and creating navigation elements such as bookmarks or thumbnail images;
  - copying or otherwise extracting text and graphics from the document;
  - filling out forms (that is, filling out existing interactive form fields) and signing the document (which amounts to filling out existing signature fields, a type of interactive form field);
  - modifying the contents of the document;
  - printing the document.

Note: The earliest version of the PDF file which matches the specified properties of the PDFEncryptionInfo object is selected as the version of the PDF file.

- The earliest file version available is version 1.3.
- If at least one of the AllowFillingFormFields, AllowExtractingTextAndGraphicsExt, AllowDocumentAssembling or AllowPrintingExt properties is TRUE, or the EncryptionLevel property is PEL_High, the PDF file version will be 1.4.
- If the EncryptionLevel property is PEL_HighAES, the version will be 1.6.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AllowAddingTextAnnotations</td>
<td>Boolean</td>
<td>Enables/disables adding or modifying text annotations and interactive form fields. The default value is FALSE. Enables/disables assembling the document: inserting, rotating, or deleting pages and creating navigation elements such as bookmarks or thumbnail images. The default value is FALSE.</td>
</tr>
<tr>
<td>AllowDocumentAssembling</td>
<td>Boolean</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>


AllowExtractingTextAndGraphics **Boolean**

Enables/disables copying or otherwise extracting text and graphics from the document. The default value is FALSE.

AllowExtractingTextAndGraphicsExt **Boolean**

Enables/disables extracting text and graphics (to make the document accessible to users with disabilities or for other purposes). The default value is FALSE.

AllowFillingFormFields **Boolean**

Enables/disables filling out forms (that is, filling out existing interactive form fields) and signing the document (which amounts to filling out existing signature fields, a type of interactive form field). The default value is FALSE.

AllowModifyingContent **Boolean**

Enables/disables modifying the contents of the document. The default value is FALSE.

AllowPrinting **Boolean**

Enables/disables printing the document. The default value is FALSE.

AllowPrintingExt **Boolean**

Enables/disables printing to a representation from which a faithful digital copy of the PDF content could be generated. Disallowing such printing may result in degradation of output quality (a feature implemented as "Print As Image" in Acrobat). The default value is FALSE.

EncryptionLevel **PDFEncryptionLevelEnum**

Sets the level of encryption. The default value is PEL_Low.

OwnerPassword **String**

Stores the owner password. Opening the document with the correct owner password (assuming it is not the same as the user password) allows full (owner) access to the document. This unlimited
access includes the ability to change the document’s passwords and access permissions. The default value is an empty string.

Stores the user password. Opening the document with the correct user password (or opening a document that does not have a user password) allows additional operations to be performed according to the user access permissions specified in the document’s encryption dictionary. The default value is an empty string.

See also

IPDFExportSettings::EncryptionInfo
ABBY Recognition Server 4 COM-based API
HeaderAndFooterSettings Object
(IHeaderAndFooterSettings Interface)

This object provides functionality for tuning header and footer parameters when exporting recognized text to PDF (PDF/A) format. These parameters are set in the HeaderAndFooter property of the PDFExportSettings (PDFAExportSettings) object. The HeaderAndFooterSettings object allows you to do the following:

- Place headers and footers on each document page;
- Specify the text to be stamped, the location of the header or footer on the page, and other formatting options;
- Include into the header or footer any static text, as well as page numbers, Bates numbers (auto-incrementing number), current date and time, output file name, etc. To include a variable into the footer or header text, insert a suitable tag from the list below:
  - `<BatesNum>` Auto-incrementing numbers
  - `<Dd>` Current date
  - `<EmailSubject>` Subject of the incoming e-mail message
  - `<FileName>` Name of the output file
  - `<Folder>` Name of parent subfolder in image folder
  - `<Mm>` Current month
  - `<PageNum>` Page number
  - `<Yy>` Current year (2 digits)
  - `<Yyyy>` Current year (4 digits)

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BottomMargin</td>
<td>String</td>
<td>Sets the bottom margin. The value of this property must be in the range of 0 to 120 inches (0 to 3048 millimeters). The default value is 0,5 inches.</td>
</tr>
<tr>
<td>CentralFooter</td>
<td>String</td>
<td>Specifies the central footer text. The default value is an empty string.</td>
</tr>
<tr>
<td>CentralHeader</td>
<td>String</td>
<td>Specifies the central header text. The default value is an empty string.</td>
</tr>
<tr>
<td>FontName</td>
<td>String</td>
<td>Sets the font name. The default value is “Times New Roman”.</td>
</tr>
<tr>
<td>FontSize</td>
<td>String</td>
<td>Sets the font size in points. The default value is 12.</td>
</tr>
<tr>
<td>Property</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>--------</td>
<td>-----------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>IsBold</td>
<td>String</td>
<td>Specifies if the bold font style has to be used. The default value is FALSE.</td>
</tr>
<tr>
<td>IsInInches</td>
<td>String</td>
<td>Specifies whether measurement units for the margins are inches. If the value of this property is FALSE, the measurement units for the margins are millimeters. The default value is TRUE.</td>
</tr>
<tr>
<td>IsItalic</td>
<td>Long</td>
<td>Specifies if the italic font style has to be used. The default value is FALSE.</td>
</tr>
<tr>
<td>IsUnderlined</td>
<td>Boolean</td>
<td>Specifies if the underlined font style has to be used. The default value is FALSE.</td>
</tr>
<tr>
<td>LeftFooter</td>
<td>Boolean</td>
<td>Specifies the left footer text. The default value is an empty string.</td>
</tr>
<tr>
<td>LeftHeader</td>
<td>Boolean</td>
<td>Specifies the left header text. The default value is an empty string.</td>
</tr>
<tr>
<td>LeftMargin</td>
<td>Long</td>
<td>Sets the left margin. The value of this property must be in the range of 0 to 120 inches (0 to 3048 millimeters). The default value is 1 inch.</td>
</tr>
<tr>
<td>NumberofDigits</td>
<td>Double</td>
<td>Sets the number of digits in the Bates numbers. If the number of digits in a Bates number is less than specified by this property, the corresponding number of zeros will be added in front of the Bates number. The maximal allowed number of digits is 9. The default value is 5.</td>
</tr>
<tr>
<td>RightFooter</td>
<td>Double</td>
<td>Specifies the right footer text. The default value is an empty string.</td>
</tr>
<tr>
<td>RightHeader</td>
<td>Double</td>
<td>Specifies the right header text. The default value is an empty string.</td>
</tr>
<tr>
<td>RightMargin</td>
<td>Double</td>
<td>Sets the right margin. The value of this property must be in the range of 0 to 120 inches (0 to 3048 millimeters). The default value is 1 inch.</td>
</tr>
<tr>
<td>StartingNumber</td>
<td>Boolean</td>
<td>Sets the starting Bates number. The number of digits in the starting number must not exceed the number specified in the NumberofDigits property. The default value is 1.</td>
</tr>
<tr>
<td>TextColor</td>
<td>Long</td>
<td>Sets the text color in RGB format. The default value is red or RGB(255, 0, 0). Note: The Long value is calculated from the RGB triplet using the formula: ((red\ value) + (256 \times green\ value) + (65536 \times blue\ value)), where red value is the first triplet component, green value is the second triplet component, blue value is the third triplet component. Hence the Long value of the color red equals 255.</td>
</tr>
<tr>
<td>TopMargin</td>
<td>Long</td>
<td>Sets the top margin. The value of this property must be in the range of 0 to 120 inches (0 to 3048 millimeters). The default value is 0,5 inches.</td>
</tr>
</tbody>
</table>

See also

IPDFExportSettings::HeaderAndFooter,
IPDFAExportSettings::HeaderAndFooter
ABBYY Recognition Server 4 COM-based API
SMTPSettings Object (ISMTPSettings Interface)

This object allows to set the parameters for connection with the SMTP e-mail server. These parameters are set in the SMTPSettings property of the OutputFormatSettings object.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Password</td>
<td>String</td>
<td>Specifies the password corresponding to the username. Depending on SMTP server settings the password may not be required. The default value is an empty string.</td>
</tr>
<tr>
<td>PortNumber</td>
<td>Long</td>
<td>Specifies the network port number of the SMTP server. The default value is 25.</td>
</tr>
<tr>
<td>SenderAddress</td>
<td>String</td>
<td>Specifies the e-mail address of the output message sender. The default value is an empty string.</td>
</tr>
<tr>
<td>ServerHostName</td>
<td>String</td>
<td>Specifies the SMTP server address. The default value is an empty string.</td>
</tr>
<tr>
<td>ServerRequiresAuthentication</td>
<td>Boolean</td>
<td>Specifies if authentication is required on the SMTP server. If the value of this property is FALSE, Recognition Server will try to determine whether authentication is required automatically. The default value is FALSE. Note: ABBYY Recognition Server only supports SMTP servers that use the PLAIN authentication method.</td>
</tr>
</tbody>
</table>

See also

IOutputFormatSettings::SMTPSettings
ABBYY Recognition Server 4 COM-based API
**DocumentInfoItems Object**  
*(IDocumentInfoItems Interface)*

This object provides access to a collection of `DocumentInfoItem` objects which represent metadata from the PDF (PDF/A) file. A reference to this object is returned by the `IPDFExportSettings::DocumentInfo` (`IPDFAExportSettings::DocumentInfo`) property.

You can add key-value pairs to the `DocumentInfoItems` collection using the `Add` method. The key name (`IDocumentInfoItem::Name`) must not be an empty string and must not repeat any of the other key names of the collection.

You must not add key-value pairs with Creator and Producer key names. The value of the Producer key name can be set in the `IPDFExportSettings::Producer` (`IPDFAExportSettings::Producer`) property. The value of the Creator key name cannot be changed. It is the same as in the source document.

You must not add key-value pairs with ModDate and CreationDate key names. The value of the ModDate and CreationDate key names cannot be changed. The value of the ModDate key is set to the current date, the value of the CreationDate key is copied from the source document, or is set to the current date.

If you set the values of the Author, Keywords, Subject or Title keys, the values of the corresponding properties of the `PDFExportSettings` (`PDFAExportSettings`) object will be set to the same value. Vice versa, if you change the values of the Author, Keywords, Subject or Title properties of the `PDFExportSettings` (`PDFAExportSettings`) object, the values of the corresponding keys will be changed or the corresponding key-value pairs will be added to the collection.

**Properties**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Count</strong></td>
<td><code>Long</code></td>
<td>Stores the number of elements in the collection.</td>
</tr>
</tbody>
</table>

**Methods**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Add</strong></td>
<td>Adds a new <code>DocumentInfoItem</code> object at the end of the collection.</td>
</tr>
<tr>
<td><strong>Item</strong></td>
<td>Provides access to a single element of the collection.</td>
</tr>
<tr>
<td><strong>Remove</strong></td>
<td>Removes an element from the collection.</td>
</tr>
<tr>
<td><strong>RemoveAll</strong></td>
<td>Removes all the elements from the collection.</td>
</tr>
</tbody>
</table>
See also

Working with Collections, DocumentInfoItem, IPDFExportSettings::DocumentInfo, IPDFAExportSettings::DocumentInfo
ABBYY Recognition Server 4 COM-based API
DocumentInfoItem Object
(IDocumentInfoItem Interface)

This object contains metadata from the PDF (PDF/A) file. It represents a key-value pair.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>String</td>
<td>Stores the name of metadata which is used as a key.</td>
</tr>
<tr>
<td>Value</td>
<td>String</td>
<td>Stores the value of metadata.</td>
</tr>
</tbody>
</table>

See also

DocumentInfoItems
ABBYY Recognition Server 4 COM-based API
**AltoMeasurementUnitEnum**

`AltoMeasurementUnitEnum` enumeration constants are used to set the measurement unit used to describe size and coordinates of objects in the output XML file.

```c
typedef enum {
    AMU_Inch1200,
    AMU_Mm10,
    AMU_Pixel
} AltoMeasurementUnitEnum;
```

**Elements**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AMU_Inch1200</td>
<td>The unit of measure is inch/1200.</td>
</tr>
<tr>
<td>AMU_Mm10</td>
<td>The unit of measure is mm/10.</td>
</tr>
<tr>
<td>AMU_Pixel</td>
<td>The unit of measure is pixel.</td>
</tr>
</tbody>
</table>

**See also**

[AltoExportSettings](#)
ABBY Recognition Server 4 COM-based API
AttributeTypeEnum

**AttributeTypeEnum** enumeration constants define different types of document attributes.

typedef enum {
    AT_Boolean,
    AT_Enumeration,
    AT_SingleLine,
    AT_MultipleLines
    AT_RegularExpression
} AttributeTypeEnum;

**Elements**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT_Boolean</td>
<td>Checkbox.</td>
</tr>
<tr>
<td>AT_Enumeration</td>
<td>Element of the list.</td>
</tr>
<tr>
<td>AT_SingleLine</td>
<td>Single line.</td>
</tr>
<tr>
<td>AT_MultipleLines</td>
<td>Multiple lines.</td>
</tr>
<tr>
<td>AT_RegularExpression</td>
<td>Regular expression.</td>
</tr>
</tbody>
</table>

**See also**

[DocumentAttribute](#)
ABBYY Recognition Server 4 COM-based API
BarcodeTypeEnum

**BarcodeTypeEnum** enumeration constants are used to specify the type of barcode used for separating documents.

typedef enum {
    IBT_Code39,
    IBT_CheckCode39,
    IBT_Code39WithoutAsterisk,
    IBT_Interleaved25,
    IBT_CheckInterleaved25,
    IBT_Ean13,
    IBT_Code128,
    IBT_Ean8,
    IBT_Pdf417,
    IBT_Codabar,
    IBT_Upce,
    IBT_Industrial25,
    IBT_Iata25,
    IBT_Matrix25,
    IBT_Code93,
    IBT_Postnet,
    IBT_Ucc128,
    IBT_Patch,
    IBT_Upca,
    IBT_Aztec,
    IBT_Datamatrix,
    IBT_Qrcode,
    IBT_MaxiCode,
    IBT_Code32,
    IBT_FullASCII,
    IBT_Royal,
    IBT_Kix,
    IBT_Intelligent,
    IBT_Unknown
} BarcodeTypeEnum;

**Elements**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBT_Aztec</td>
<td>Aztec is a high density two-dimensional matrix style barcode that can encode up to 3750 characters from the entire 256 byte ASCII character set. The symbol is built on a square grid with a bulls-eye pattern at its center.</td>
</tr>
<tr>
<td>IBT_CheckCode39</td>
<td>Check Code 39 is a Code 39 barcode with a check sum.</td>
</tr>
<tr>
<td>IBT_CheckInterleaved25</td>
<td>Check Interleaved 2 of 5 is an Interleaved 2 of 5 barcode with a check sum.</td>
</tr>
<tr>
<td>IBT_Codabar</td>
<td>Codabar is a self-checking, variable length barcode that can encode 16 data characters. It is used primarily for numeric data, but also</td>
</tr>
</tbody>
</table>
**IBT_Codabar**

encodes six special characters. Codabar is useful for encoding dollar and mathematical figures because a decimal point, plus sign, and minus sign can be encoded.

**IBT_Code128**

Code 128 is an alphanumeric, very high-density, compact, variable length barcode scheme that can encode the full 128 ASCII character set. Each character is represented by three bars and three spaces totaling 11 modules. Each bar or space is one, two, three, or four modules wide with the total number of modules representing bars an even number and the total number of modules representing a space an odd number. Three different start characters are used to select one of three character sets.

**IBT_Code39**

Code 39, also referred to as Code 3 of 9, is an alphanumeric, self-checking, variable length barcode that uses five black bars and four spaces to define a character. Three of the elements are wide and six are narrow.

Code 39 without asterisk is a Code 39 barcode, which has no start and stop symbol, the asterisk "*.

**IBT_Code39WithoutAsterisk**

*Note: In ABBYY Recognition Serve, barcodes of type **Code 39 without asterisk** can only be used for document separation, as they are recognized only if the user has explicitly specified their type.

**IBT_Code93**

Code 93 is a variable length barcode that encodes 47 characters. It is named Code 93 because every character is constructed from nine elements arranged into three bars with their adjacent spaces. Code 93 is a compressed version of Code 39 and was designed to complement Code 39.

**IBT_Datamatrix**

Data Matrix is a two-dimensional matrix barcode consisting of black and white modules arranged in either a square or rectangular pattern. Every Data Matrix is composed of two solid adjacent borders in an "L" shape and two other borders consisting of alternating dark and light modules. Within these borders are rows and columns of cells encoding information. A Data Matrix barcode can store up to 2335 alphanumeric characters.

**IBT_Ean8**

The European Article Numbering (EAN) system is used for products that require a country origin. This is a fixed-length barcode used to encode either eight or thirteen characters. The first two characters identify the country of origin, the next characters are data characters, and the last character is the checksum.

**IBT_Ean13**

**IBT_IATA25**

IATA 2 of 5 is a numeric, variable length barcode. It is a barcode standard designed by the IATA (International Air Transport Association). This standard is used for all boarding passes.

**IBT_Industrial25**

Industrial 2 of 5 is numeric-only barcode that has been in use a long time. Unlike Interleaved 2 of 5, all of the information is encoded in the bars; the spaces are fixed width and are used only to separate the
# Barcodes

## IBT_Intelligent

Intelligent Mail Barcode (IM barcode) is a barcode that encodes up to 31 digits of information about the route and delivery means of a mail piece. It contains 65 vertical bars of four types, each of which consists of a central portion (a bar that contains this portion only is also referred to as a tracker), and may include a top portion (ascender), a bottom portion (descender) or both (full bar). The 65 bars represent ten 13-bit characters, totaling 130 bits. Each of these characters contains 2, 5, 8 or 11 bits set to one, and the Hamming distance between characters is at least 2. Due to this, single-bit errors in characters can be automatically detected and corrected. The Intelligent Mail Barcode has been adopted by the United States Postal Services.

## IBT_Interleaved25

Interleaved 2 of 5 is a variable length (must be a multiple of two), high-density, self-checking, numeric barcode that uses five black bars and five white bars to define a character. Two digits are encoded in every character; one in the black bars and one in the white bars. Two of the black bars and two of the white bars are wide. The other bars are narrow.

## IBT_Matrix25

Standard 2 of 5 is self-checking numeric-only barcode. Unlike Interleaved 2 of 5, all of the information is encoded in the bars; the spaces are fixed width and are used only to separate the bars. Matrix 2 of 5 is used primarily for warehouse sorting, photo finishing, and airline ticket marking.

## IBT_Patch

Patch Code is 1 character long barcode, which does not encode data, but acts as a signal. It is used only for batch separation and scanner control.

## IBT_PDF417

PDF417 is a variable length, two-dimensional (2D), stacked symbology that can store up to 1850 printable ASCII characters or 1100 binary characters per symbol. PDF417 is designed with selectable levels of error correction. Its high data capacity can be helpful in applications where a large amount of data must travel with a labeled document or item.

## IBT_Postnet

The Postnet (Postal Numeric Encoding Technique) is a fixed length symbology (5, 6, 9, or 11 characters) which uses constant bar and space width. Information is encoded by varying the bar height between the two values. Postnet barcodes are placed on the lower right of envelopes or postcards, and are used to expedite the processing of mail with automatic equipment and provide reduced postage rates.

## IBT_Qrcode

QR Code is a two-dimensional matrix barcode. The barcode has 3 large squares (registration marks) in the corners which define the top of the barcode. The black and white squares in the area between the registration marks are the encoded data and error correction keys.
QR Codes can encode over 4000 ASCII characters.

**IBT_Ucc128**

This type of barcode is a 19 digit barcode with a 20th check digit. For a total of 20 digits. It typically is used for carton identification. Both for internal carton numbering and also for using the UCC-128 barcode on your cartons being shipped out to your customers.

If this value is specified, the program will use barcodes of all types to separate documents.

**IBT_Unknown**

The UPC-A (Universal Product Code) barcode is 12 digits long, including its checksum. Each digit is represented by a seven-bit sequence, encoded by a series of alternating bars and spaces. UPC-A is used for marking products which are sold at retail in the USA. Note that UPC-A codes with 2 or 5 digit supplemental codes appended to them are not supported.

The UPC-E barcode is a shortened version of UPC-A barcode. It compresses the data characters and the checksum into six characters. This bar code is ideal for small packages because it is the smallest bar code. Note that UPC-E codes with 2 or 5 digit supplemental codes appended to them are not supported.

See also

*IExportParams::SeparationMethod*
ABBYY Recognition Server 4 COM-based API
**CodePageEnum**

**CodePageEnum** enumeration is the Open API internal representation of code pages.

```csharp
enum CodePageEnum{
    CP_Null = 0,
    CP_Latin = 1252,
    CP_Cyrillic = 1251,
    CP_EasternEuropean = 1250,
    CP_Baltic = 1257,
    CP_Turkish = 1254,
    CP_US_MSDOS = 437,
    CP_LatinI_MSDOS = 850,
    CP_Russian_MSDOS = 866,
    CP_Baltic_MSDOS = 775,
    CP_Turkish_IBM = 857,
    CP_Slavic_MSDOS = 852,
    CP_Greek = 1253,
    CP_Greek_737 = 737,
    CP_Greek_869 = 869,
    CP_Latin_ISO = 28591,
    CP_EasternEuropean_ISO = 28592,
    CP_Turkish_ISO = 28593,
    CP_Baltic_ISO = 28594,
    CP_Cyrillic_ISO = 28595,
    CP_Greek_ISO = 28597,
    CP_KOI8 = 20866,
    CP_Tatar = 5000,
    CP_Tatar_MSDOS = 5001,
    CP_Roman_Macintosh = 10000,
    CP_Greek_Macintosh = 10006,
    CP_Cyrillic_Macintosh = 10007,
    CP_Ukrainian_Macintosh = 10017,
    CP_Latin2_Macintosh = 10029,
    CP_Icelandic_Macintosh = 10079,
    CP_Turkish_Macintosh = 10081,
    CP_Croatian_Macintosh = 10082,
    CP_Armenian = 5002,
    CP_Armenian_MSDOS = 5003,
    CP_Armenian_Macintosh = 5004,
    CP_Latin5_ISO = 28599,
    CP_Cyrillic_MSDOS = 855,
    CP_Bashkir = 5006
};
```

**Elements**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CP_Null</td>
<td>Invalid code page.</td>
</tr>
<tr>
<td>CP_Latin</td>
<td>Windows Western Europe (1252)</td>
</tr>
<tr>
<td>CP_Cyrillic</td>
<td>Windows Cyrillic (1251)</td>
</tr>
<tr>
<td>Code Name</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>-----------------------------------</td>
</tr>
<tr>
<td>CP_EasternEuropean</td>
<td>Windows Central European (1250)</td>
</tr>
<tr>
<td>CP_Baltic</td>
<td>Windows Baltic (1257)</td>
</tr>
<tr>
<td>CP_Turkish</td>
<td>Windows Turkish ( 1254)</td>
</tr>
<tr>
<td>CP_US_MSDOS</td>
<td>DOS United States (437)</td>
</tr>
<tr>
<td>CP_Latin1_MSDOS</td>
<td>DOS Multilingual Latin 1 (850)</td>
</tr>
<tr>
<td>CP_Russian_MSDOS</td>
<td>DOS Russian (866)</td>
</tr>
<tr>
<td>CP_Baltic_MSDOS</td>
<td>DOS Baltic (775)</td>
</tr>
<tr>
<td>CP_Turkish IBM</td>
<td>DOS Turkish (857)</td>
</tr>
<tr>
<td>CP_Slavic_MSDOS</td>
<td>DOS Latin 2 (852)</td>
</tr>
<tr>
<td>CP_Greek</td>
<td>Windows Greek (1253)</td>
</tr>
<tr>
<td>CP_Greek_737</td>
<td>DOS Greek (737)</td>
</tr>
<tr>
<td>CP_Greek_869</td>
<td>DOS Modern Greek (869)</td>
</tr>
<tr>
<td>CP_Latin_ISO</td>
<td>ISO Latin 1 (8859-1)</td>
</tr>
<tr>
<td>CP_EasternEuropean_ISO</td>
<td>ISO Central Europe (8859-2)</td>
</tr>
<tr>
<td>CP_Turkish_ISO</td>
<td>ISO Latin 3 (8859-3)</td>
</tr>
<tr>
<td>CP_Baltic_ISO</td>
<td>ISO Baltic (8859-4)</td>
</tr>
<tr>
<td>CP_Cyrillic_ISO</td>
<td>ISO Cyrillic (8859-5)</td>
</tr>
<tr>
<td>CP_Greek_ISO</td>
<td>ISO Greek (8859-7)</td>
</tr>
<tr>
<td>CP_KOI8</td>
<td>Russian KOI8</td>
</tr>
<tr>
<td>CP_Tatar</td>
<td>Windows Tatar</td>
</tr>
<tr>
<td>CP_Tatar_MSDOS</td>
<td>DOS Tatar</td>
</tr>
<tr>
<td>CP_Roman_Macintosh</td>
<td>Macintosh Roman</td>
</tr>
<tr>
<td>CP_Greek_Macintosh</td>
<td>Macintosh Greek 1</td>
</tr>
<tr>
<td>CP_Cyrillic_Macintosh</td>
<td>Macintosh Cyrillic</td>
</tr>
<tr>
<td>CP_Ukrainian_Macintosh</td>
<td>Macintosh Ukraine</td>
</tr>
<tr>
<td>CP_Latin2_Macintosh</td>
<td>Macintosh Latin 2</td>
</tr>
<tr>
<td>CP_Icelandic_Macintosh</td>
<td>Macintosh Icelandic</td>
</tr>
<tr>
<td>CP_Turkish_Macintosh</td>
<td>Macintosh Turkish</td>
</tr>
<tr>
<td>CP_Croatian_Macintosh</td>
<td>Macintosh Croatian</td>
</tr>
<tr>
<td>CP_Armenian</td>
<td>Windows Armenian</td>
</tr>
<tr>
<td>CP_Armenian_MSDOS</td>
<td>DOS Armenian</td>
</tr>
<tr>
<td>CP_Armenian_Macintosh</td>
<td>Macintosh Armenian</td>
</tr>
<tr>
<td>CP_Latin5_ISO</td>
<td>ISO Turkish (8859-9)</td>
</tr>
<tr>
<td>CP_Cyrillic_MSDOS</td>
<td>DOS Cyrillic</td>
</tr>
<tr>
<td>CP_Bashkir</td>
<td>Bashkir</td>
</tr>
</tbody>
</table>

See also
ICSVExportSettings::CodePage,
IHTMLExportSettings::CodePage,
ITextExportSettings::CodePage
ABBY Recognition Server 4 COM-based API
**DocumentSeparationMethodEnum**

*DocumentSeparationMethodEnum* enumeration constants are used to set the document separation method during export.

```c
typedef enum {
    DSM_OneFilePerImage,
    DSM_ByNumberOfPages,
    DSM_ByBlankPages,
    DSM_BySubfolder,
    DSM_ByBarcode,
    DSM_MergeIntoSingleFile
} DocumentSeparationMethodEnum;
```

### Elements

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSM_OneFilePerImage</td>
<td>The output file will have the same number of pages as the input file.</td>
</tr>
<tr>
<td></td>
<td>A new file is created for every N pages received by the server. This</td>
</tr>
<tr>
<td></td>
<td>constant can not be used as the input parameter of the</td>
</tr>
<tr>
<td></td>
<td><strong>IExportParams::SeparationMethod</strong> property. It can be received only</td>
</tr>
<tr>
<td></td>
<td>as the output parameter of this property.</td>
</tr>
<tr>
<td>DSM_ByNumberOfPages</td>
<td>A new file will be created at each blank page found in the document</td>
</tr>
<tr>
<td></td>
<td>flow. This constant can not be used as the input parameter of the</td>
</tr>
<tr>
<td></td>
<td><strong>IExportParams::SeparationMethod</strong> property. It can be received only</td>
</tr>
<tr>
<td></td>
<td>as the output parameter of this property.</td>
</tr>
<tr>
<td>DSM_ByBlankPages</td>
<td>The files in each subfolder will be merged into one document. This</td>
</tr>
<tr>
<td></td>
<td>constant can not be used as the input parameter of the</td>
</tr>
<tr>
<td></td>
<td><strong>IExportParams::SeparationMethod</strong> property. It can be received only</td>
</tr>
<tr>
<td></td>
<td>as the output parameter of this property.</td>
</tr>
<tr>
<td>DSM_BySubfolder</td>
<td>A new file will be created at each page with a barcode of the specified</td>
</tr>
<tr>
<td></td>
<td>types. This constant can not be used as the input parameter of the</td>
</tr>
<tr>
<td></td>
<td><strong>IExportParams::SeparationMethod</strong> property. It can be received only</td>
</tr>
<tr>
<td></td>
<td>as the output parameter of this property.</td>
</tr>
<tr>
<td>DSM_ByBarcode</td>
<td>The input files will be merged into one document.</td>
</tr>
<tr>
<td>DSM_MergeIntoSingleFile</td>
<td></td>
</tr>
</tbody>
</table>

**See also**

**IExportParams::SeparationMethod**
ABBYY Recognition Server 4 COM-based API
**EPUBVersionEnum** enumeration constants specify different EPUB versions.

typedef enum {
    EV_2_0_1,
    EV_3
} EPUBVersionEnum;

**Elements**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EV_2_0_1</td>
<td>EPUB version 2.0.1.</td>
</tr>
<tr>
<td>EV_3</td>
<td>EPUB version 3.</td>
</tr>
</tbody>
</table>

**See also**

EPUBExportSettings::EPUBVersion
### ExportPictureFormatEnum

**ExportPictureFormatEnum** enumeration constants are used to set the color mode and compression method for images inside the PDF (PDF/A) and EPUB files.

```java
enum ExportPictureFormatEnum{
    EPF_Automatic,
    EPF_Ccitt4,
    EPF_JpegColor,
    EPF_JpegGray,
    EPF_PngColor,
    EPF_PngGray,
    EPF_LzwColor,
    EPF_LzwGray,
    EPF_ZipColor,
    EPF_ZipGray,
    EPF_J2KColor,
    EPF_J2KGray,
    EPF_JBIG2
};
```

### Elements

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EPF_Automatic</td>
<td>The compression method and color mode are selected automatically based on two image properties: Color Type (black and white, grayscale, or color) and Color Variety (low or high). Black-and-white images are always saved using the CCITT4 compression algorithm. Grayscale and color images are saved using the ZIP compression algorithm in the case of low color variety, and using the JPEG compression algorithm in the case of high color variety.</td>
</tr>
<tr>
<td>EPF_Ccitt4</td>
<td>This format allows you to save pictures in black-and-white mode. The CCITT4 compression algorithm is used.</td>
</tr>
<tr>
<td>EPF_JpegColor</td>
<td>Images are saved in color JPEG format. This format is suitable for documents containing color scanned or digital photos.</td>
</tr>
<tr>
<td>EPF_JpegGray</td>
<td>Images are saved in gray JPEG format. This format is suitable for scanned or digital photos saved in gray-scale mode.</td>
</tr>
<tr>
<td>EPF_PngColor</td>
<td>Images are saved in color PNG format. This format is suitable for documents containing color scanned or digital photos.</td>
</tr>
<tr>
<td>EPF_PngGray</td>
<td>Images are saved in gray PNG format. This format is suitable for scanned or digital photos saved in gray-scale mode.</td>
</tr>
<tr>
<td>EPF_LzwColor</td>
<td>Images are saved in color LZW format. This format is suitable for graphics. This format is not available for PDF/A files.</td>
</tr>
<tr>
<td>EPF_LzwGray</td>
<td>Images are saved in gray LZW format. This format is suitable for graphics and gray images. This format is not available for PDF/A files.</td>
</tr>
<tr>
<td>EPF_ZipColor</td>
<td>Images are saved in color ZIP format. This format is suitable for pictures created</td>
</tr>
</tbody>
</table>
with paint programs.

Images are saved in gray ZIP format. This format allows you to save pictures in gray-scale mode.

Images are saved in color JPEG 2000 format. This format is suitable for documents containing color scanned or digital photos. This format is not available for PDF/A files.

Images are saved in gray JPEG 2000 format. This format is suitable for scanned or digital photos saved in gray-scale mode. This format is not available for PDF/A files.

Images are saved in gray JBIG2 format. This format is suitable for binary images.

See also

IPDFExportSettings::PictureFormat,
IPDFAEExportSettings::PictureFormat,
IEPUBExportSettings::PictureFormat
ABBYY Recognition Server 4 COM-based API
FontAttributesModeEnum

FontAttributesModeEnum enumeration constants are used to set the layout retention method for files saved in EPUB format.

typedef enum {
    FFM_Full,
    FFM_Plain,
    FFM_Restricted
} FontAttributesModeEnum;

Elements

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FFM_Full</td>
<td>Produced document maintains the formatting of the original.</td>
</tr>
<tr>
<td>FFM_Plain</td>
<td>Text formatting is not preserved.</td>
</tr>
<tr>
<td>FFM_Restricted</td>
<td>Retains fonts, font sizes, and paragraphs, but does not retain the exact locations of the objects on the page or the spacing. The resulting text will be left-aligned.</td>
</tr>
</tbody>
</table>

See also

IEPUBExportSettings::FontFormattingMode
ABBYY Recognition Server 4 COM-based API
**HTMLSynthesisModeEnum**

**HTMLSynthesisModeEnum** enumeration constants are used to define available modes of synthesizing of HTML code from the recognized text.

typedef enum {
    HSM_PlainText,
    HSM_FormattedStream,
    HSM_PageLayout
} HTMLSynthesisModeEnum;

**Elements**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HSM_PlainText</td>
<td>Only paragraphs are retained in the recognized text by using of the <code>&lt;p&gt;</code> tag.</td>
</tr>
<tr>
<td>HSM_FormattedStream</td>
<td>Paragraphs and fonts of the recognized text are retained in the output HTML file. The <code>&lt;p&gt;</code> tag is used.</td>
</tr>
<tr>
<td>HSM_PageLayout</td>
<td>Full layout of the input page is retained by using a table.</td>
</tr>
</tbody>
</table>

**See also**

*IHTMLExportSettings::HTMLSynthesisMode*
ABBYY Recognition Server 4 COM-based API
ImageColorModeEnum

ImageColorModeEnum enumeration constants are used to set the image color.

typedef enum {
    ICM_AsIs,
    ICM_Color,
    ICM_Gray,
    ICM_BlackAndWhite
} ImageColorModeEnum;

Elements

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ICM_AsIs</td>
<td>Image will be exported as is.</td>
</tr>
<tr>
<td>ICM_Color</td>
<td>Color image will be exported.</td>
</tr>
<tr>
<td>ICM_Gray</td>
<td>Gray image will be exported.</td>
</tr>
<tr>
<td>ICM_BlackAndWhite</td>
<td>Black and white image will be exported.</td>
</tr>
</tbody>
</table>

See also

ITiffExportSettings::ColorMode, IJpegExportSettings::ColorMode, IJpeg2kExportSettings::ColorMode
ABBYY Recognition Server 4 COM-based API
ImageCompressionTypeEnum enumeration constants are used to set the compression files.

typedef enum {
    ICT_Uncompressed,
    ICT_Zip,
    ICT_Group3,
    ICT_Group4,
    ICT_PackBits,
    ICT_Jpeg,
    ICT_Lzw
} ImageCompressionTypeEnum;

Elements

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ICT_Uncompressed</td>
<td>Image will be exported without compression</td>
</tr>
<tr>
<td>ICT_Zip</td>
<td>Image will be exported using the ZIP compression algorithm</td>
</tr>
<tr>
<td>ICT_Group3</td>
<td>Image will be exported using the GROUP3 compression algorithm</td>
</tr>
<tr>
<td>ICT_Group4</td>
<td>Image will be exported using the GROUP4 compression algorithm</td>
</tr>
<tr>
<td>ICT_PackBits</td>
<td>Image will be exported using the PACKBITS compression algorithm</td>
</tr>
<tr>
<td>ICT_Jpeg</td>
<td>Image will be exported using the JPEG compression algorithm</td>
</tr>
<tr>
<td>ICT_Lzw</td>
<td>Image will be exported using the LZW compression algorithm.</td>
</tr>
</tbody>
</table>

See also

ITiffExportSettings::Compression
ABBY Recognition Server 4 COM-based API
JobStateEnum

TextTypeEnum enumeration constants are used to describe the type of recognized text.

typedef enum {
    JS_NoSuchJob,
    JS_Waiting,
    JS_Paused,
    JS_Processing,
    JS_Verification,
    JS_VerificationWait,
    JS_Indexing,
    JS_IndexingWait,
    JS_Processed,
    JS_ProcessedPaused,
    JS_Publishing
} JobStateEnum;

Elements

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JS_NoSuchJob</td>
<td>There is no job with specified ID.</td>
</tr>
<tr>
<td>JS_Waiting</td>
<td>The job is waiting to be processed.</td>
</tr>
<tr>
<td>JS_Paused</td>
<td>The job is in the <strong>Paused</strong> state if corresponding workflow was stopped based on a schedule.</td>
</tr>
<tr>
<td>JS_Processing</td>
<td>The job is being processed.</td>
</tr>
<tr>
<td>JS_Verification</td>
<td>One or several pages of the job are being verified.</td>
</tr>
<tr>
<td>JS_VerificationWait</td>
<td>All pages of the job are waiting to be verified.</td>
</tr>
<tr>
<td>JS_Indexing</td>
<td>Document of the job is being indexed.</td>
</tr>
<tr>
<td>JS_IndexingWait</td>
<td>Document of the job is waiting to be indexed.</td>
</tr>
<tr>
<td>JS_Processed</td>
<td>The job has been processed, but has not yet been published.</td>
</tr>
<tr>
<td>JS_ProcessedPaused</td>
<td>The job cannot be published.</td>
</tr>
<tr>
<td>JS_Publishing</td>
<td>The job is being published.</td>
</tr>
</tbody>
</table>

See also

IClient::GetJobState
OutputFileFormatEnum enumeration constants define different file formats in which ABBYY Recognition Server can save.

typedef enum {
    OFF_MSWord,
    OFF_MSExcel,
    OFF_RTF,
    OFF_XML,
    OFF_PDF,
    OFF_PDFA,
    OFF_Text,
    OFF_CSV,
    OFF_HTML,
    OFF_NoConversion,
    OFF_TIFF,
    OFF_JPG,
    OFF_J2K,
    OFF_InternalFormat,
    OFF_DOCX,
    OFF_XLSX,
    OFF_JBIG2,
    OFF_AltoXML
} OutputFileFormatEnum;

Elements

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OFF_MSWord</td>
<td>Microsoft Word format. The parameters of a file in this format are tuned through the MSWordExportSettings object.</td>
</tr>
<tr>
<td>OFF_MSExcel</td>
<td>Microsoft Excel format. The parameters of a file in this format are tuned through the XLExportSettings object.</td>
</tr>
<tr>
<td>OFF_RTF</td>
<td>Rich text format. The parameters of a file in this format are tuned through the RTFExportSettings object.</td>
</tr>
<tr>
<td>OFF_XML</td>
<td>XML format. The parameters of a file in this format are tuned through the XMLExportSettings object.</td>
</tr>
<tr>
<td>OFF_PDF</td>
<td>PDF format. The parameters of a file in this format are tuned through the PDFExportSettings object.</td>
</tr>
<tr>
<td>OFF_PDFA</td>
<td>PDF/A format. The parameters of a file in this format are tuned through the PDFAExportSettings object.</td>
</tr>
<tr>
<td>OFF_Text</td>
<td>Text format. The parameters of a file in this format are tuned through the TextExportSettings object.</td>
</tr>
<tr>
<td>OFF_CSV</td>
<td>CSV format. The parameters of a file in this format are tuned through the CSVExportSettings object.</td>
</tr>
<tr>
<td>OFF_HTML</td>
<td>HTML format. The parameters of a file in this format are tuned through the</td>
</tr>
</tbody>
</table>


OFF_HTML  HTMLExportSettings object.
OFF_NoConversion  Saves input file without conversion.
OFF_TIFF  TIFF format. The parameters of a file in this format are tuned through the TiffExportSettings object.
OFF_JPG  JPEG format. The parameters of a file in this format are tuned through the JpegExportSettings object.
OFF_J2K  JPEG 2000 format. The parameters of a file in this format are tuned through the Jpeg2kExportSettings object.
OFF_InternalFormat  ABBYY FineReader internal format.
OFF_DOCX  Microsoft Word 2007 format. The parameters of a file in this format are tuned through the DOCXExportSettings object.
OFF_XLSX  Microsoft Excel 2007 format. The parameters of a file in this format are tuned through the XLSXExportSettings object.
OFF_JBIG2  JBIG2 format. The parameters of a file in this format are tuned through the JBig2ExportSettings object.
OFF_AltoXML  AltoXML format. The parameters of a file in this format are tuned through the AltoExportSettings object.

See also
IClient::CreateOutputFormatSettings, OutputFormatSettings
ABBYY Recognition Server 4 COM-based API
OutputPublishingMethodEnum enumeration constants are used to specify the destination of the output documents.

typedef enum {
    OPM_SharedFolder,
    OPM_SharePoint,
    OPM_Mail,
    OPM_SMTP
} OutputPublishingMethodEnum;

Elements

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPM_SharedFolder</td>
<td>The output files are published to the shared folder.</td>
</tr>
<tr>
<td>OPM_SharePoint</td>
<td>The output files are published to the Microsoft SharePoint document library</td>
</tr>
<tr>
<td>OPM_Mail</td>
<td>The output files are sent to the e-mail addresses via Exchange server.</td>
</tr>
<tr>
<td>OPM_Smtp</td>
<td>The output files are sent to the e-mail addresses via SMTP server.</td>
</tr>
</tbody>
</table>

**Note:** ABBYY Recognition Server only supports SMTP servers that use the PLAIN authentication method.

See also

IOutputFormatSettings::PublishingMethod
ABBYY Recognition Server 4 COM-based API
PDFAModeEnum

PDFAModeEnum enumeration constants are used to set the PDF/A standard.

typedef enum {
    PDFAM_PdfA_1a,
    PDFAM_PdfA_1b,
    PDFAM_PdfA_2a,
    PDFAM_PdfA_2b,
    PDFAM_PdfA_2u,
    PDFAM_PdfA_3a,
    PDFAM_PdfA_3b,
    PDFAM_PdfA_3u
} PDFAModeEnum;

Elements

PDF/A-1a — Level A compliance in Part 1
PDF/A-1b — Level B compliance in Part 1
PDF/A-2a — Level A compliance in Part 2
PDF/A-2b — Level B compliance in Part 2
PDF/A-2u — Level B compliance in Part 2 with an additional requirement that all text in the document have Unicode mapping
PDF/A-3a — Level A compliance in Part 3
PDF/A-3b — Level B compliance in Part 3
PDF/A-3u — Level B compliance in Part 3 with an additional requirement that all text in the document have Unicode mapping

See also

IPDFAExportSettings::Mode
ABBYY Recognition Server 4 COM-based API
PDFEncryptionLevelEnum enumeration constants are used to set the level of encryption of the PDF file during export.

```c
typedef enum {
    PEL_Low,
    PEL_High,
    PEL_HighAES
} PDFEncryptionLevelEnum;
```

**Elements**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PEL_Low</td>
<td>Sets a low (40-bit) encryption level.</td>
</tr>
<tr>
<td>PEL_High</td>
<td>Sets a high (128-bit) encryption level, but Acrobat 3.0 users cannot open PDF documents with this encryption level.</td>
</tr>
<tr>
<td>PEL_HighAES</td>
<td>Sets a high (128-bit AES) encryption level, but Acrobat 6.0 (or earlier) users cannot open PDF documents with this encryption level.</td>
</tr>
</tbody>
</table>

**See also**

IPDFEncryptionInfo::EncryptionLevel
PDFExportModeEnum

PDFExportModeEnum enumeration constants are used to set the mode of export into PDF format.

typedef enum {
    PEM_TextAndPictures,
    PEM_TextOnImage,
    PEM_ImageOnText,
    PEM_ImageOnly
} PDFExportModeEnum;

Elements

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PEM_TextAndPictures</td>
<td>The recognized text is saved as text, and the pictures are saved as pictures. The original document design (font, background, and layout marking) is not retained.</td>
</tr>
<tr>
<td>PEM_TextOnImage</td>
<td>The entire image is saved as a picture. Text areas are saved as text over the image.</td>
</tr>
<tr>
<td>PEM_ImageOnText</td>
<td>The entire image is saved as a picture. The recognized text is put under it. This option is useful if you export your text to document archives: the full page layout is retained and the full-text search is available if you save in this mode.</td>
</tr>
<tr>
<td>PEM_ImageOnly</td>
<td>The entire image is saved as a picture. The recognized text and layout information are not used in this mode, so the recognition stage may be skipped.</td>
</tr>
</tbody>
</table>

See also

IPDFExportSettings::ExportMode
ABBYY Recognition Server 4 COM-based API
PDFVersionEnum

PDFVersionEnum enumeration constants specify different PDF versions.

typedef enum {
    PVN_Auto,
    PVN_Version13,
    PVN_Version14,
    PVN_Version15,
    PVN_Version16,
    PVN_Version17
} PDFVersionEnum;

Elements

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PVN_Auto</td>
<td>The version is detected automatically.</td>
</tr>
<tr>
<td>PVN_Version13</td>
<td>PDF version 1.3.</td>
</tr>
<tr>
<td>PVN_Version14</td>
<td>PDF version 1.4.</td>
</tr>
<tr>
<td>PVN_Version15</td>
<td>PDF version 1.5.</td>
</tr>
<tr>
<td>PVN_Version16</td>
<td>PDF version 1.6.</td>
</tr>
<tr>
<td>PVN_Version17</td>
<td>PDF version 1.7.</td>
</tr>
</tbody>
</table>

See also

PDFExportSettings::PDFVersion,
PDFAExportSettings::PDFVersion
ABBYY Recognition Server 4 COM-based API


**PictureColorControlTypeEnum** enumeration constants are used to set color mode for output PDF file.

typedef enum {
    PCCT_Auto,
    PCCT_ColorToBwBinary,
    PCCT_ColorToGray,
    PCCT_SaveColoration
} ImageColorModeEnum;

**Elements**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCCT_Auto</td>
<td>Automatic.</td>
</tr>
<tr>
<td>PCCT_ColorToBwBinary</td>
<td>Convert color and grayscale images to black-and-white using binarization.</td>
</tr>
<tr>
<td>PCCT_ColorToGray</td>
<td>Convert color images to grayscale.</td>
</tr>
<tr>
<td>PCCT_SaveColoration</td>
<td>Keep original color mode.</td>
</tr>
</tbody>
</table>

See also

[PDFExportSettings](#), [PDFAExportSettings](#)
ABBYY Recognition Server 4 COM-based API
**PriorityEnum**

**PriorityEnum** enumeration constants are used to set priorities of the jobs.

define enum {
    P_Low,
    P_BelowNormal,
    P_Normal,
    P_AboveNormal,
    P_High
} PriorityEnum;

**Elements**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>P_Low</td>
<td>Low job priority.</td>
</tr>
<tr>
<td>P_BelowNormal</td>
<td>Below normal job priority.</td>
</tr>
<tr>
<td>P_Normal</td>
<td>Normal job priority.</td>
</tr>
<tr>
<td>P_AboveNormal</td>
<td>Above normal job priority.</td>
</tr>
<tr>
<td>P_High</td>
<td>High job priority.</td>
</tr>
</tbody>
</table>

See also

IXmlTicket::Priority,
IXmlResult::Priority
ABBYY Recognition Server 4 COM-based API
RecognitionModeEnum

RecognitionModeEnum enumeration constants are used to set the mode of document recognition.

typedef enum {
    RM_FullPage,
    RM_BarcodesOnly
} RecognitionModeEnum;

Elements

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RM_FullPage</td>
<td>All document content is recognized including text inside tables, pictures and barcodes (if the IRecognitionParams::LookForBarcodes property is set to TRUE).</td>
</tr>
<tr>
<td>RM_BarcodesOnly</td>
<td>Only barcode values are extracted from the document.</td>
</tr>
</tbody>
</table>

See also

IRecognitionParams::RecognitionMode
ABBYY Recognition Server 4 COM-based API
RecognitionQualitySettingEnum

RecognitionQualitySettingEnum enumeration constants are used to set the recognition mode.

typedef enum {
    RQS_Thorough,
    RQS_Balanced,
    RQS_Fast
} RecognitionQualitySettingEnum;

Elements

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RQS_Thorough</td>
<td>The recognition will run in full mode which is optimized for quality. The recognition will run in balanced mode (an intermediate mode between full and fast modes, which specify with the help of the RQS_Thorough and RQS_Fast constants).</td>
</tr>
<tr>
<td>RQS_Balanced</td>
<td></td>
</tr>
<tr>
<td>RQS_Fast</td>
<td>The recognition will run in fast mode which provides 2-2.5 times faster recognition speed at the cost of a moderately increased error rate (1.5-2 times more errors). On good print quality texts, ABBYY Recognition Server makes an average of 1-2 errors per page, and such moderate increase in error rate can be easily tolerated in many cases, such as full text indexing with &quot;fuzzy&quot; searches, preliminary recognition, etc.</td>
</tr>
</tbody>
</table>

See also

RecognitionParams
RotationType Enum

RotationType Enum enumeration constants are used to set the types of rotation that to be performed upon an image.

typedef enum {
    RT_NoRotation,
    RT_Automatic,
    RT_Clockwise,
    RT_Counterclockwise,
    RT_Upsidedown
} RotationTypeEnum;

Elements

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RT_NoRotation</td>
<td>This value means no rotation.</td>
</tr>
<tr>
<td>RT_Automatic</td>
<td>Rotation will be detected automatically.</td>
</tr>
<tr>
<td>RT_Clockwise</td>
<td>Rotation 90 degrees clockwise.</td>
</tr>
<tr>
<td>RT_Counterclockwise</td>
<td>Rotation 90 degrees counterclockwise or 270 degrees clockwise.</td>
</tr>
<tr>
<td>RT_Upsidedown</td>
<td>Rotation upside down or 180 degrees.</td>
</tr>
</tbody>
</table>

See also

IPreprocessingParams::RotationType. See sample: Sample for ASP.NET.
ABBYY Recognition Server 4 COM-based API
**RTFSynthesisModeEnum**

 enumeration constants are used to set the modes of RTF file synthesis from the recognized text when exporting to RTF format.

```c
typedef enum {
    RSM_PlainText,
    RSM_FormattedText,
    RSM_EditableCopy,
    RSM_ExactCopy
} RTFSynthesisModeEnum;
```

**Elements**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RSM_PlainText</td>
<td>Unlike the <strong>RSM_FormattedText</strong> mode, this mode does not retain formatting. Retains fonts, font sizes, and paragraphs, but does not retain the exact locations of the objects on the page or the spacing. The resulting text will be left-aligned. (Right-to-left texts will be right-aligned). <strong>Note.</strong> Vertical texts will be changed to horizontal in this mode.</td>
</tr>
<tr>
<td>RSM_FormattedText</td>
<td>Produces a document that nearly preserves the original format and text flow but allows easy editing.</td>
</tr>
<tr>
<td>RSM_EditableCopy</td>
<td>Produces a document that maintains the formatting of the original. This option is recommended for documents with complex layouts, such as promotion booklets. Note, however, that this option limits the ability to change the text and formatting of the output document.</td>
</tr>
<tr>
<td>RSM_ExactCopy</td>
<td></td>
</tr>
</tbody>
</table>

**See also**

[IRTFExportSettings::RTFSynthesisMode](#), [IMSWordExportSettings::RTFSynthesisMode](#)
ABYY Recognition Server 4 COM-based API
TextCoordinatesParticularityEnum

textCoordinatesParticularityEnum enumeration constants are used to specify how text should be divided: by words or by lines.

typedef enum {
    TCP_Lines,
    TCP_Words
} TextCoordinatesParticularityEnum;

Elements

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCP_Lines</td>
<td>The text should be divided by lines.</td>
</tr>
<tr>
<td>TCP_Words</td>
<td>The text should be divided by words.</td>
</tr>
</tbody>
</table>

See also

AltoExportSettings
ABBYY Recognition Server 4 COM-based API
**TextEncodingTypeEnum**

**TextEncodingTypeEnum** enumeration is the Open API internal representation of text encoding types.

```c
typedef enum {
    TET_Simple,
    TET_UTF8,
    TET_UTF16
} TextEncodingTypeEnum;
```

### Elements

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TET_Simple</td>
<td>Simple encoding.</td>
</tr>
<tr>
<td>TET_UTF8</td>
<td>Unicode UTF-8 format.</td>
</tr>
<tr>
<td>TET_UTF16</td>
<td>Unicode UTF-16 format.</td>
</tr>
</tbody>
</table>

**See also**

- `ICSVExportSettings::EncodingType`
- `ITextExportSettings::EncodingType`
- `IHTMLExportSettings::EncodingType`
ABBYY Recognition Server 4 COM-based API
TextTypeEnum

TextTypeEnum enumeration constants are used to describe the type of recognized text.

typedef enum {
    TT_Normal,
    TT_Typewriter,
    TT_Matrix,
    TT_OCR_A,
    TT_OCR_B,
    TT_MICR_E13B,
    TT_Gothic,
    TT_Fax
} TextTypeEnum;

Elements

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TT_Normal</td>
<td>This value corresponds to common typographic text.</td>
</tr>
<tr>
<td>TT_Typewriter</td>
<td>This value tells Open API to presume that the text on the image was typed on a typewriter.</td>
</tr>
<tr>
<td>TT_Matrix</td>
<td>This value tells Open API to presume that the text on the image was printed by means of a dot-matrix printer.</td>
</tr>
<tr>
<td>TT_OCR_A</td>
<td>This value corresponds to a monospaced font designed specifically for Optical Character Recognition. Largely used by banks, credit card companies and similar businesses. It may look as follows:</td>
</tr>
<tr>
<td></td>
<td><strong>OCR A 123</strong></td>
</tr>
<tr>
<td>TT_OCR_B</td>
<td>This value corresponds to a font designed specifically for Optical Character Recognition. It may look as follows:</td>
</tr>
<tr>
<td></td>
<td><strong>OCR B 123</strong></td>
</tr>
<tr>
<td>TT_MICR_E13B</td>
<td>This value corresponds to a special set of numeric characters printed with special magnetic inks. MICR (Magnetic Ink Character Recognition) characters are found in a variety of places, including personal checks. It may look as follows:</td>
</tr>
<tr>
<td></td>
<td>0123456789.</td>
</tr>
</tbody>
</table>

Note: If an image contains characters of a language and MICR characters, you must specify the MICR (E13B) text type and select the E13B language in addition to the main recognition language. You need not select the E13B language when recognizing English texts, because the English language already contains all E13B
characters.
This value tells Open API to presume that the text on the image was printed in Gothic type. It may look as follows:

TT_Gothic

Die Verwahrung gegen

For this text type, Open API currently supports the following fonts: Textura (Gothic script), Fraktur and Schwabacher.

TT_Fax

This value tells Open API to presume that the text on the recognized image is produced by a fax machine.

See also

IRecognitionParams::TextTypes
ABBYY Recognition Server 4 COM-based API
**VerificationModeEnum**

`VerificationModeEnum` enumeration constants are used to set the type of verification.

**Elements**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DVM_DoNotVerify</td>
<td>Verification is turned off: documents will skip the verification stage.</td>
</tr>
<tr>
<td>DVM_VerifyAlways</td>
<td>Verification is turned: documents will always go through the verification stage.</td>
</tr>
<tr>
<td>DVM_VerifyIfThresholdExceeded</td>
<td>Documents will be submitted to the verification stage if the verification threshold set in <code>VerificationModeThreshold</code> is exceeded.</td>
</tr>
</tbody>
</table>

**See also**

`VerificationModeThreshold`
ABBYY Recognition Server 4 COM-based API
**WorkflowStateEnum**

WorkflowStateEnum enumeration constants are used to set the state of a workflow.

**Elements**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WS_ApplyingSettings</td>
<td>The state of a workflow after it has been started and before the processing has begun. At this stage, the program checks if it can access the folder that contains the input documents. This state is very short in duration and is not indicated in the console (the word &quot;Starting&quot; is displayed instead).</td>
</tr>
<tr>
<td>WS_Crawling</td>
<td>At this stage, the program checks the folders of the Document Library workflow. It counts the files, adds them to the database, and prepares to process them. The word &quot;Crawling&quot; is displayed in the console.</td>
</tr>
<tr>
<td>WS_Finishing</td>
<td>The state of a workflow when processing is coming to an end. At this stage, the program writes the files for the last time and completes publishing the large files. The words &quot;Finishing Processing&quot; are displayed in the console.</td>
</tr>
<tr>
<td>WS_NotAvailable</td>
<td>The state of a workflow that is inaccessible. The words &quot;Not Available&quot; are displayed in the console, together with the reason why the workflow cannot be accessed.</td>
</tr>
<tr>
<td>WS_Processing</td>
<td>The principal state of a workflow, when files are being received, processed, and recognized. The word &quot;Processing&quot; is displayed in the console.</td>
</tr>
<tr>
<td>WS_StartingProcess</td>
<td>The state of a workflow after the start command has been executed and before information about the beginning of processing has been returned. The word &quot;Starting&quot; is displayed in the console.</td>
</tr>
<tr>
<td>WS_Suspended</td>
<td>The state of a workflow that has been stopped. The word &quot;Stopped&quot; is displayed in the console.</td>
</tr>
</tbody>
</table>

**See also**

Workflow
Working with Properties

The Web Services API objects have various properties and methods. There are some important things to remember when working with object properties in the Web Services API. When using SOAP through the object model of VisualStudio only a wrapper for an object is created on the client side, i.e. a class is created which has only data fields. This class itself cannot verify data integrity, so data integrity will only be verified when processing the request on the Web server. Also, data fields on default are at the discretion of compiler. First, a wrapper for the object is created which has no error indication, then it is sent to the server with the request, and only after that an error message can be received in the server response. Therefore you need to set all the property fields of the objects instead of relying on the default values of these properties.

The Web Services API properties may be of the following types:

<table>
<thead>
<tr>
<th>Type</th>
<th>C# type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>bool (with two values true and false)</td>
</tr>
<tr>
<td>Integer</td>
<td>int</td>
</tr>
<tr>
<td>String</td>
<td>string</td>
</tr>
<tr>
<td>Byte Array</td>
<td>byte[]</td>
</tr>
<tr>
<td>Enumeration</td>
<td>enum</td>
</tr>
</tbody>
</table>
ABBYY Recognition Server 4 Web Services API
Working with Collections

**Important:** The indexing of ABBYY Recognition Server Open API collections starts with 0.

Collection classes are implemented for several types of objects in the Open API. The following collection types are available:

- **StringsCollection** — a collection of strings
- **OutputFormatSettingsCollection** — a collection of **OutputFormatSettings** objects
- **InputFiles** — a collection of **InputFile** objects
- **OutputDocuments** — a collection of **OutputDocument** objects
- **Workflows** — a collection of **Workflow** objects
- **DocumentInfoItems** — a collection of **DocumentInfoItem** objects
- **DocumentAttributes** — a collection of **DocumentAttribute** objects
- **DocumentTypes** — a collection of **DocumentType** objects
- **IndexingFields** — a collection of **IndexingField** objects

These collections are independent objects and are used to pass various sets of parameters to functions that require them.

All these objects are characterized by a common set of properties and methods that make them collections.

These properties and methods are (in IDL notation):

### Standard collection-specific properties and methods:

```idl
// This property stores the number of elements in the collection
HRESULT Count([out, retval] long* pVal);

// This method provides access to a single collection element
HRESULT Item([in] long index, [out, retval]<collection type>* pVal);
```

### ABBYY Recognition Server collection-specific properties and methods (optional):

```idl
// Inserts a new element at the specified position
HRESULT Insert([in]<collection type> newVal, [in] long index);

// Adds a new element at the end of the collection
HRESULT Add([in]<collection type> newVal);

// Removes an element from the collection
HRESULT Remove([in] long index);

// Removes all elements from the collection
HRESULT RemoveAll();
```

See also

See samples: [Hello](#), [Listening](#), [AsyncProcessing](#), [Sample for ASP.NET](#).
Using the Web Services API within 64-bit ASP.NET

The ABBYY Recognition Server 4 Web Service cannot be used directly within 64-bit ASP.NET. To use the Web Service within 64-bit web application framework create a COM+ application using the Recognition Server COM-based API. (See Using COM-based API within 64-bit applications for details.) After the COM+ application is created, the Recognition Server Web Service can be used within 64-bit ASP.NET.

See also

Using the Web API
Using the Web API

In order to use Web API:

- COM-based API has to be installed and properly configured at the computer with ABBYY Recognition Server. (See Using COM-based API within 64-bit applications for details.)
- Internet Information Services version 7 or higher, .NetFramework v.3.5.SP1 have to be installed at computer with ABBYY Recognition Server.

The installing program of ABBYY Recognition Server automatically configures system to use COM API.

For troubleshooting purposes see the steps that are required for manual configuration:

1. The specific Application Pool should be created in IIS to host ABBYY Recognition Server Web API application. The required properties of this application pool are:
   - Name = ABBYY Recognition Server 4;
   - .NetFramework version = v.2.0 (.NetFramework 3.5SP1 should be installed at the computer);
   - Managed Pipeline Mode = Integrated;
   - Identity = «LocalSystem» by default, or user that was specified in wizard of installation program. It is important that the Application pool should be configured to work under the same user account, that is used for running ABBYY Recognition Server service (Select Start > Control Panel > Administrative Tools > Services and see ABBYY Recognition Server 4 Server Manager service).

2. The specific virtual application should be created from the folder [ABBYY Recognition Server 4 installation folder]\RecognitionWS, called Recognition4WS. The properties of the application are:
   - Application Pool = «ABBYY Recognition Server 4», that was created in first step;
   - Virtual path = /Recognition4WS

To perform these steps or just to observe the actual settings Select Start > Control Panel > Administrative Tools > Internet Information Services (IIS) Manager.

See also

Using the Web Services API within 64-bit ASP.NET
ABBYY Recognition Server 4 Web Services API
RSSoapService Object

This object allows you to exchange of XML-based messages with ABBYY Recognition Server to process recognition jobs.

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DoNothingMethod</td>
<td>This is a service method, you should not use it.</td>
</tr>
<tr>
<td>GetWorkflows</td>
<td>Returns a set of available workflows on the Recognition Server with specified location. Only workflows with the following types of Input Folder are available: shared folder, FTP folder.</td>
</tr>
<tr>
<td>ProcessFile</td>
<td>Uses the parameters of the specified workflow to recognize the specified input image.</td>
</tr>
<tr>
<td>CreateTicket</td>
<td>Creates an XmlTicket object based on the specified workflow.</td>
</tr>
<tr>
<td>ProcessTicket</td>
<td>Uses the parameters of the XmlTicket object and the specified workflow to recognize the input image.</td>
</tr>
<tr>
<td>DeleteJob</td>
<td>Completely deletes the job.</td>
</tr>
<tr>
<td>GetJobState</td>
<td>Allows you to get information about current state of the specified job.</td>
</tr>
<tr>
<td>StartProcessFile</td>
<td>Uses the parameters of the specified workflow to recognize the input image in asynchronous mode.</td>
</tr>
<tr>
<td>StartProcessTicket</td>
<td>Uses the parameters of the XmlTicket and the specified workflow to recognize the input image in asynchronous mode.</td>
</tr>
</tbody>
</table>

See also

XmlTicket. See sample: WebServiceHello.
ABBYY Recognition Server 4 Web Services API
XmlTicket Object

This object represents the processing parameters of one job. It can be created by using the CreateTicket method of the RSSoapService object based on one workflow from the list returned by the RSSoapService::GetWorkflows method.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attributes</td>
<td>DocumentAttributes</td>
<td>Stores the DocumentAttributes object, where you can set document attributes.</td>
</tr>
<tr>
<td>ExportParams</td>
<td>ExportParams</td>
<td>Stores the ExportParams object, where you can set export parameters.</td>
</tr>
<tr>
<td>InputFiles</td>
<td>InputFile Array</td>
<td>Stores the image set.</td>
</tr>
<tr>
<td>Name</td>
<td>String</td>
<td>Stores the name of XML Ticket, which will be used when saving the results.</td>
</tr>
<tr>
<td>PreprocessingParams</td>
<td>PreprocessingParams</td>
<td>Stores the PreprocessingParams object, where you can set image processing parameters.</td>
</tr>
<tr>
<td>Priority</td>
<td>PriorityEnum</td>
<td>Sets the priority of the job.</td>
</tr>
<tr>
<td>RecognitionParams</td>
<td>RecognitionParams</td>
<td>Stores the RecognitionParams object, where you can set recognition parameters.</td>
</tr>
<tr>
<td>UserProperty</td>
<td>String</td>
<td>Stores any user-defined string. The string is passed to the UserProperty property of the XmlResult object.</td>
</tr>
</tbody>
</table>

Input parameter

The XmlTicket object is the input parameter of the RSSoapService::ProcessTicket method.

Output parameter

The XmlTicket object is the output parameter of the RSSoapService::CreateTicket method.

See also

XmlResult, XML Ticket description. See sample: WebServiceHello.
ABBYY Recognition Server 4 Web Services API
XmlResult Object

This object represents the parameters and results of processing one job.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Errors</td>
<td>String Array</td>
<td>Returns a list of errors that occurred when processing the job.</td>
</tr>
<tr>
<td>ExportParams</td>
<td>ExportParams</td>
<td>Stores the export parameters</td>
</tr>
<tr>
<td>InputFiles</td>
<td>InputFile Array</td>
<td>Stores the set of image files and processing results.</td>
</tr>
<tr>
<td>JobDocuments</td>
<td>JobDocument Array</td>
<td>Stores the set of job documents.</td>
</tr>
<tr>
<td>IsFailed</td>
<td>Boolean</td>
<td>If an error occurred during processing, this property will be set to TRUE.</td>
</tr>
<tr>
<td>Name</td>
<td>String</td>
<td>Stores the name of XML Result, which coincides with the name of the XML Ticket that was used for recognition.</td>
</tr>
<tr>
<td>PreprocessingParams</td>
<td>PreprocessingParams</td>
<td>Stores the image processing parameters.</td>
</tr>
<tr>
<td>Priority</td>
<td>PriorityEnum</td>
<td>Stores the priority of the job.</td>
</tr>
<tr>
<td>RecognitionParams</td>
<td>RecognitionParams</td>
<td>Stores the recognition parameters.</td>
</tr>
<tr>
<td>Statistics</td>
<td>Statistics, read-only</td>
<td>Returns the job processing statistics.</td>
</tr>
<tr>
<td>UserProperty</td>
<td>String</td>
<td>Stores the user-defined string which is specified the UserProperty property of the XmlTicket object.</td>
</tr>
<tr>
<td>Warnings</td>
<td>String Array</td>
<td>Returns a list of warnings that were issued when processing the job.</td>
</tr>
</tbody>
</table>

Output parameter

The XmlResult object is the output parameter of the RSSoapService::ProcessFile and RSSoapService::ProcessTicket methods.

See also

XmlTicket, XML Result description. See sample: WebServiceHello.
ABBYY Recognition Server 4 Web Services API
InputFile Object

This object represents one input image file and the results of processing this file.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attributes</td>
<td>DocumentAttributes, read-only</td>
<td>The property is retained for backward compatibility. For getting the document attributes, use the Attributes property of the JobDocument object.</td>
</tr>
<tr>
<td>CustomText</td>
<td>String, read-only</td>
<td>The property is retained for backward compatibility. For getting the custom text, use the CustomText property of the JobDocument object.</td>
</tr>
<tr>
<td>Errors</td>
<td>StringsCollection, read-only</td>
<td>Returns a list of errors that occurred when processing the image.</td>
</tr>
<tr>
<td>FileName</td>
<td>String</td>
<td>Stores the name of the image file. Note: You cannot include any subfolders into the file name.</td>
</tr>
<tr>
<td>ID</td>
<td>String, read-only</td>
<td>Stores the unique identifier of the input file.</td>
</tr>
<tr>
<td>OutputDocuments</td>
<td>OutputDocuments, read-only</td>
<td>Returns a collection of output files of different formats belonging to the document.</td>
</tr>
<tr>
<td>Pages</td>
<td>Pages, read-only</td>
<td>Returns a collection of pages in the input image file.</td>
</tr>
<tr>
<td>Password</td>
<td>String</td>
<td>Stores the password for accessing the PDF file. The default value is an empty string.</td>
</tr>
<tr>
<td>Statistics</td>
<td>Statistics, read-only</td>
<td>Returns image processing statistics.</td>
</tr>
<tr>
<td>Warnings</td>
<td>StringsCollection, read-only</td>
<td>Returns a list of warnings that were issued when processing the image.</td>
</tr>
</tbody>
</table>

See also

InputFiles.
ABBYY Recognition Server 4 Web Services API
JobDocuments Object

This object represents a collection of JobDocument objects.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count</td>
<td>Long</td>
<td>Stores the number of elements in the collection.</td>
</tr>
</tbody>
</table>

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add</td>
<td>Adds a new element at the end of the collection.</td>
</tr>
<tr>
<td>Item</td>
<td>Provides access to a single element of the collection.</td>
</tr>
<tr>
<td>Remove</td>
<td>Removes an element from the collection.</td>
</tr>
<tr>
<td>RemoveAll</td>
<td>Removes all the elements from the collection.</td>
</tr>
</tbody>
</table>

See also

JobDocument, XmlTicket, XmlResult, Working with Collections.
JobDocument Object (IJobDocument Interface)

This object represents one input image file and the results of processing this file.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attributes</td>
<td>DocumentAttributes, read-only</td>
<td>Stores document attributes.</td>
</tr>
<tr>
<td>BarcodeText</td>
<td>String, read-only</td>
<td>Returns the text of the separation barcode.</td>
</tr>
<tr>
<td>CustomText</td>
<td>String, read-only</td>
<td>Custom text assigned to the document during document separation or indexing. The text can be used for output file naming. You can include this text to the output file name using the &lt;CustomText&gt; tag.</td>
</tr>
<tr>
<td>Errors</td>
<td>StringsCollection, read-only</td>
<td>Returns a list of errors that occurred when processing the document.</td>
</tr>
<tr>
<td>Name</td>
<td>String</td>
<td>Stores the name of the job document. The name is generated automatically.</td>
</tr>
<tr>
<td>OutputDocuments</td>
<td>OutputDocuments, read-only</td>
<td>Returns a collection of output files of different formats belonging to the document.</td>
</tr>
<tr>
<td>PagePositions</td>
<td>PagePositions, read-only</td>
<td>Returns a collection of the output document positions.</td>
</tr>
<tr>
<td>Statistics</td>
<td>Statistics, read-only</td>
<td>Returns document processing statistics.</td>
</tr>
<tr>
<td>Warnings</td>
<td>StringsCollection, read-only</td>
<td>Returns a list of warnings that were issued when processing the document.</td>
</tr>
</tbody>
</table>

See also

JobDocuments.
ABBYY Recognition Server 4 Web Services API
FileContainer Object

This object represents one input image file and the results of processing this file.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FileContents</td>
<td>Byte Array</td>
<td>Stores the file contents.</td>
</tr>
<tr>
<td>OpenPassword</td>
<td>String</td>
<td>Stores the password for accessing the PDF file.</td>
</tr>
<tr>
<td>LocationPath</td>
<td>String</td>
<td>Stores the UNC-path to the file.</td>
</tr>
</tbody>
</table>

Input parameter

The FileContainer object is the input parameter of the RSSoapService::ProcessFile method.

See also

InputFile::FileData. See sample: WebServiceHello.
**PreprocessingParams Object**

This object specifies how an image will be preprocessed before analysis and recognition.

**Properties**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConvertToBWFormat</td>
<td>Boolean</td>
<td>Specifies if the image must be converted to black and white (B/W) during preprocessing. <strong>Note:</strong> If this property is set to TRUE, the speed of recognition will increase, but the quality will decrease.</td>
</tr>
<tr>
<td>CorrectResolution</td>
<td>Boolean</td>
<td>Specifies if the image resolution must be corrected. The default value is TRUE.</td>
</tr>
<tr>
<td>Deskew</td>
<td>Boolean</td>
<td>Specifies if the skew angle for an image must be corrected during preprocessing. This mode is recommended if you want to correct skew for images you work with.</td>
</tr>
<tr>
<td>RemoveGarbage</td>
<td>Boolean</td>
<td>Specifies if garbage (excess dots that are smaller than a certain size) must be removed from the image during preprocessing.</td>
</tr>
<tr>
<td>RemoveTexture</td>
<td>Boolean</td>
<td>Specifies if background noise must be cleared before the process starts.</td>
</tr>
<tr>
<td>RotationType</td>
<td>RotationTypeEnum</td>
<td>Specifies what type of rotation will be performed upon the preprocessing.</td>
</tr>
<tr>
<td>SplitDualPages</td>
<td>Boolean</td>
<td>Specifies if the dual pages must be split during preprocessing.</td>
</tr>
</tbody>
</table>

**See also**

(XmlTicket::PreprocessingParams, XmlResult::PreprocessingParams)
ABBYY Recognition Server 4 Web Services API
RecognitionParams Object

This object allows you to tune the recognition parameters.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Languages</td>
<td>String array</td>
<td>Specifies a set of recognition languages. Elements of this set must be internal names from the first column of the ABBYY Recognition Server recognition languages.</td>
</tr>
<tr>
<td>LookForBarcodes</td>
<td>Boolean</td>
<td>Specifies whether barcodes must be recognized.</td>
</tr>
<tr>
<td>RecognitionMode</td>
<td>RecognitionModeEnum</td>
<td>Specifies the recognition mode.</td>
</tr>
<tr>
<td>RecognitionQuality</td>
<td>RecognitionQualitySettingEnum</td>
<td>Specifies the mode which optimizes recognition for quality or speed. The value of this property is an OR superposition of the TextType Enum. The value of this property must not be an empty set.</td>
</tr>
<tr>
<td>TextTypes</td>
<td>Integer</td>
<td></td>
</tr>
</tbody>
</table>

See also

XmlTicket::RecognitionParams, XmlResult::RecognitionParams
DocumentAttributes Object

This object represents a list of DocumentAttribute objects.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Items</td>
<td>DocumentAttribute_Array</td>
<td>Stores a set of document attributes.</td>
</tr>
<tr>
<td>DocumentType</td>
<td>String</td>
<td>Specifies a document type.</td>
</tr>
<tr>
<td>SkipManualIndexing</td>
<td>Boolean</td>
<td>Specifies whether manual indexing is to be skipped. If the value of this property is FALSE, the document will be queued for processing at the Indexing Station.</td>
</tr>
</tbody>
</table>

See also

XmlTicket, DocumentAttribute
ABBYY Recognition Server 4 Web Services  API
DocumentAttribute Object

This object represents a document attribute. It may be typecast to one of its child objects:

- **BooleanAttribute**
- **EnumerationAttribute**
- **SingleLineAttribute**
- **MultipleLinesAttribute**
- **RegularExpressionAttribute**

These objects allow access to values of different types and inherit the properties of the DocumentAttribute object. They are also elements of the DocumentAttributes list.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>String, read-only</td>
<td>Attribute name.</td>
</tr>
<tr>
<td>Type</td>
<td>AttributeTypeEnum, read-only</td>
<td>Attribute type.</td>
</tr>
</tbody>
</table>

See also

DocumentAttributes
ABBYY Recognition Server 4 Web Services API
BooleanDocumentAttribute Object

This object provides access to the value of the Checkbox attribute. The BooleanDocumentAttribute is a child object of the DocumentAttribute and inherits its properties. It is an element of the DocumentAttributes list.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>Boolean</td>
<td>This property sets the value of the Checkbox attribute.</td>
</tr>
</tbody>
</table>

See also

DocumentAttribute, DocumentAttributes, AttributeTypeEnum
ABBYY Recognition Server 4 Web Services API
This object provides access to the value of the List attribute. The `EnumerationDocumentAttribute` is a child object of the `DocumentAttribute` and inherits its properties. It is an element of the `DocumentAttributes` list.

**Properties**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>String</td>
<td>This property specifies an element of the list.</td>
</tr>
</tbody>
</table>

**See also**

`DocumentAttribute`, `DocumentAttributes`, `AttributeTypeEnum`
SingleLineDocumentAttribute Object

This object provides access to the value of the Single line attribute. The SingleLineDocumentAttribute is a child object of the DocumentAttribute and inherits its properties. It is an element of the DocumentAttributes list.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>String</td>
<td>This property contains a single line of text as a value of the attribute.</td>
</tr>
</tbody>
</table>

See also

DocumentAttribute, DocumentAttributes, AttributeTypeEnum
**MultipleLinesDocumentAttribute Object**

This object provides access to the value of the Multiple lines attribute. The `MultipleLinesAttribute` is a child object of the [DocumentAttribute](#) and inherits its properties. It is an element of the [DocumentAttributes](#) list.

**Properties**

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>String</td>
<td>This property contains multiple lines of text as a value of the attribute.</td>
</tr>
</tbody>
</table>

See also

[DocumentAttribute](#), [DocumentAttributes](#), [AttributeTypeEnum](#)
This object provides access to the value of the Regular expression attribute. The `RegularExpressionDocumentAttribute` is a child object of the `DocumentAttribute` and inherits its properties. It is an element of the `DocumentAttributes` list.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>String</td>
<td>This property contains a regular expression as a value of the attribute.</td>
</tr>
</tbody>
</table>

See also

- `DocumentAttribute`
- `DocumentAttributes`
- `AttributeTypeEnum`
ABYY Recognition Server 4 Web Services API
ExportParams Object

This object provides functionality for tuning of export parameters of recognized text.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Formats</td>
<td>OutputFormatSettings Array</td>
<td>Stores a set of export parameters.</td>
</tr>
<tr>
<td>SeparationMethod</td>
<td>DocumentSeparationMethodEnum</td>
<td>Specifies the document separation method during export.</td>
</tr>
<tr>
<td>ResultLocationPath</td>
<td>String</td>
<td>Stores the UNC-path to the output folder.</td>
</tr>
<tr>
<td>BarcodeType</td>
<td>BarcodeTypeEnum</td>
<td>Specifies the barcode type used for document separation.</td>
</tr>
</tbody>
</table>

See also

XmlTicket::ExportParams, XmlResult::ExportParams.

See sample: WebServiceHello.
ABBYY Recognition Server 4 Web Services API
OutputDocuments Object

This object provides access to a collection of OutputDocument objects which represent output files of different formats belonging to the document. A reference to this object is returned by the IInputFile::OutputDocuments property.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count</td>
<td>Long,</td>
<td>Stores the number of elements in the collection.</td>
</tr>
</tbody>
</table>

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Item</td>
<td>Provides access to a single element of the collection.</td>
</tr>
</tbody>
</table>

See also

Working with Collections, OutputDocument, InputFile
# OutputDocument Object

This object represents parameters of files of a particular format belonging to the document.

## Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Files</td>
<td>FileContainer array</td>
<td>Stores a set of output files.</td>
</tr>
<tr>
<td>FormatSettings</td>
<td>OutputFormatSettings</td>
<td>Stores export parameters.</td>
</tr>
</tbody>
</table>

## See also

[InputFile::OutputDocuments](#)
ABBYY Recognition Server 4 Web Services API
OutputFormatSettings Object

This object provides functionality for tuning the export parameters of recognized text. It may be typecast to one of its child objects:

- MSWordExportSettings
- RTFExportSettings
- DOCXExportSettings
- XLExportSettings
- XLSXExportSettings
- XMLExportSettings
- PDFExportSettings
- PDFAExportSettings
- EPUBExportSettings
- TextExportSettings
- CSVExportSettings
- HTMLExportSettings
- TiffExportSettings
- JpegExportSettings
- Jpeg2kExportSettings
- JBig2ExportSettings
- AltoExportSettings
- NoConversionExportSettings (does not provide any settings)
- InternalFormatSettings (does not provide any settings)

These objects allow access to export parameters of different formats and inherit all the properties of the OutputFormatSettings object.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FileFormat</td>
<td>OutputFileFormatEnum</td>
<td>Returns the output file format. Do not specify the value of this property manually.</td>
</tr>
</tbody>
</table>

See also

- ExportParams::Formats,
- OutputDocument::FormatSettings.

See sample: WebServiceHello.
ABBYY Recognition Server 4 Web Services API
RTFExportSettings Object

This object provides functionality for tuning export parameters when exporting recognized text to RTF format. The RTFExportSettings object is a child object of the OutputFormatSettings object and inherits all its properties.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ForcePageSize</td>
<td>Boolean</td>
<td>Specifies whether export result must fit the dimensions set by PaperWidth and PaperHeight properties.</td>
</tr>
<tr>
<td>HighlightErrorsWithBackgroundColor</td>
<td>Boolean</td>
<td>Specifies if uncertainly recognized symbols are highlighted with background color when exported in RTF format.</td>
</tr>
<tr>
<td>PaperHeight</td>
<td>Integer</td>
<td>Stores paper height in twips (1/1440 of inch). See the table below.</td>
</tr>
<tr>
<td>PaperWidth</td>
<td>Integer</td>
<td>Stores paper width in twips (1/1440 of inch). See the table below.</td>
</tr>
<tr>
<td>RTFSynthesisMode</td>
<td>RTFSynthesisModeEnum</td>
<td>Specifies the mode of RTF file synthesis from the recognized text when exporting to RTF formats.</td>
</tr>
<tr>
<td>WritePictures</td>
<td>Boolean</td>
<td>Specifies whether pictures are to be written in files in RTF format. Note: The format in which pictures are saved in the output file is selected automatically basing on the two picture properties: Color Type (black and white, grayscale or color) and Color Variety (low or high). Black and white pictures are always saved in PNG format. Grayscale and color pictures are saved in PNG format in the case of low color.</td>
</tr>
</tbody>
</table>
variety, and in JPEG format in the case of high color variety.

Paper size in different units of measurement

<table>
<thead>
<tr>
<th>Paper size</th>
<th>in inch</th>
<th>in mm</th>
<th>in twips (1/1440 of inch)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A3</td>
<td>11,69 x 16,54</td>
<td>297 x 420</td>
<td>16838 x 23811</td>
</tr>
<tr>
<td>A4</td>
<td>8,27 x 11,69</td>
<td>210 x 297</td>
<td>11909 x 16834</td>
</tr>
<tr>
<td>A5</td>
<td>5,83 x 8,27</td>
<td>148 x 210</td>
<td>8391 x 11909</td>
</tr>
<tr>
<td>Legal</td>
<td>8,5 x 14</td>
<td>216 x 356</td>
<td>12240 x 20160</td>
</tr>
<tr>
<td>Letter</td>
<td>8,5 x 11</td>
<td>216 x 279</td>
<td>12240 x 15840</td>
</tr>
<tr>
<td>Executive</td>
<td>7,25 x 10,5</td>
<td>184 x 266</td>
<td>10440 x 15120</td>
</tr>
</tbody>
</table>

See also

[OutputFormatSettings](#)
MSWordExportSettings Object

This object provides functionality for tuning export parameters when exporting recognized text to DOC formats. The MSWordExportSettings object is a child object of the OutputFormatSettings object and inherits all its properties.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ForceFixedPageSize</td>
<td>Boolean</td>
<td>Specifies whether export result must fit the dimensions set by the PaperWidth and PaperHeight properties.</td>
</tr>
<tr>
<td>HighlightErrorsWithBackgroundColor</td>
<td>Boolean</td>
<td>Specifies if uncertainly recognized symbols are highlighted with background color when exported in DOC format.</td>
</tr>
<tr>
<td>PaperHeight</td>
<td>Integer</td>
<td>Stores paper height in twips (1/1440 of inch). The default value is the height of A4 format page. See the table &quot;Paper size in different units of measurement&quot;.</td>
</tr>
<tr>
<td>PaperWidth</td>
<td>Integer</td>
<td>Stores paper width in twips (1/1440 of inch). The default value is the width of A4 format page. See the table &quot;Paper size in different units of measurement&quot;.</td>
</tr>
<tr>
<td>RTFSynthesisMode</td>
<td>RTFSynthesisModeEnum</td>
<td>Specifies the mode of DOC file synthesis from the recognized text when exporting to DOC formats. Specifies whether pictures are to be written in files in DOC format. Note: The format in which pictures are saved in the output file is selected automatically basing on the two picture properties: Color Type (black...</td>
</tr>
</tbody>
</table>
WritePictures

Boolean

and white, grayscale or color) and Color Variety (low or high). Black and white pictures are always saved in PNG format. Grayscale and color pictures are saved in PNG format in the case of low color variety, and in JPEG format in the case of high color variety.

See also

OutputFormatSettings
**DOCXExportSettings Object**

This object provides functionality for tuning export parameters when exporting recognized text to DOCX format. The **DOCXExportSettings** object is a child object of the **OutputFormatSettings** object and inherits all its properties.

### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ForceFixedPageSize</td>
<td>Boolean</td>
<td>Specifies whether export result must fit the dimensions set by the PaperWidth and PaperHeight properties.</td>
</tr>
<tr>
<td>HighlightErrorsWithBackgroundColor</td>
<td>Boolean</td>
<td>Specifies if uncertainly recognized symbols are highlighted with background color when exported in DOCX format.</td>
</tr>
<tr>
<td>PaperHeight</td>
<td>Integer</td>
<td>Stores paper height in twips (1/1440 of inch). The default value is the height of A4 format page. See the table &quot;Paper size in different units of measurement&quot;.</td>
</tr>
<tr>
<td>PaperWidth</td>
<td>Integer</td>
<td>Stores paper width in twips (1/1440 of inch). The default value is the width of A4 format page. See the table &quot;Paper size in different units of measurement&quot;.</td>
</tr>
<tr>
<td>RTFSynthesisMode</td>
<td>RTFSynthesisModeEnum</td>
<td>Specifies the mode of DOCX file synthesis from the recognized text when exporting to DOCX formats. Specifies whether pictures are to be written in files in DOCX format. <strong>Note:</strong> The format in which pictures are saved in the output file is selected automatically basing on the two picture properties: Color Type (black</td>
</tr>
</tbody>
</table>
WritePictures [Boolean]

and white, grayscale or color) and Color Variety (low or high). Black and white pictures are always saved in PNG format. Grayscale and color pictures are saved in PNG format in the case of low color variety, and in JPEG format in the case of high color variety.

See also

OutputFormatSettings
XLExportSettings Object

This object provides functionality for tuning export parameters when exporting recognized text to XLS format. The XLExportSettings object is a child object of the OutputFormatSettings object and inherits all its properties.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConvertNumericValuesToNumbers</td>
<td>Boolean</td>
<td>Specifies whether numerical values in recognized text are exported to XLS format as numbers.</td>
</tr>
<tr>
<td>IgnoreTextOutsideTables</td>
<td>Boolean</td>
<td>If this property is set to TRUE, only text from table blocks is exported into XLS format.</td>
</tr>
</tbody>
</table>

See also

OutputFormatSettings
XLSXExportSettings Object

This object provides functionality for tuning export parameters when exporting recognized text to XLSX format. The XLSXExportSettings object is a child object of the OutputFormatSettings object and inherits all its properties.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ConvertNumericValuesToNumbers</td>
<td>Boolean</td>
<td>Specifies whether numerical values in recognized text are exported to XLSX format as numbers.</td>
</tr>
<tr>
<td>IgnoreTextOutsideTables</td>
<td>Boolean</td>
<td>If this property is set to TRUE, only text from table blocks is exported into XLSX format.</td>
</tr>
</tbody>
</table>

See also

OutputFormatSettings
**HTMLExportSettings Object**

This object provides functionality for tuning export parameters when exporting recognized text to HTML format. The **HTMLExportSettings** object is a child object of the **OutputFormatSettings** object and inherits all its properties.

### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AllowCss</td>
<td><strong>Boolean</strong></td>
<td>Specifies if a built-in style sheet (CSS) is to be used. CSS requires Internet Explorer 4.0 or later.</td>
</tr>
<tr>
<td>CodePage</td>
<td><strong>CodePageEnum</strong></td>
<td>This property sets the code page to which the recognized text is to be exported. If this property does not specify any code page (CP_Null), the code page is selected automatically.</td>
</tr>
<tr>
<td>EncodingType</td>
<td><strong>TextEncodingTypeEnum</strong></td>
<td>Specifies the encoding type of the output file in HTML format. Note: If this property is set to a value different from TET_Simple, the <strong>CodePage</strong> property is ignored during export.</td>
</tr>
<tr>
<td>HTMLSynthesisMode</td>
<td><strong>HTMLSynthesisModeEnum</strong></td>
<td>Specifies a mode of synthesizing HTML code from the recognized text. There exist three modes of synthesis: retain paragraphs only, retain paragraphs and fonts, retain full page layout.</td>
</tr>
<tr>
<td>WritePictures</td>
<td><strong>Boolean</strong></td>
<td>Specifies whether pictures must be saved along with the file in HTML format. If pictures are not written, references to them in HTML files are also omitted. Note: The format in which pictures are saved in the output file is selected automatically basing on the two picture properties: Color Type (black and white, grayscale or color) and Color Variety (low or high). Black and white pictures are always saved in PNG format. Grayscale and color pictures are saved in PNG format in the case of low color variety, and in JPEG format in the case of high color variety.</td>
</tr>
</tbody>
</table>
See also

OutputFormatSettings
XMLExportSettings Object

This object provides functionality for tuning export parameters when exporting recognized text to XML format. The XMLExportSettings object is a child object of the OutputFormatSettings object and inherits all its properties. You can see an XML scheme for an XML Document in the ExportToXml.xsd file which can be found in the Bin subfolder of the ABBYY Recognition Server folder. Also, see the Document XML Scheme and XML scheme representation for details.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PagesPerFile</td>
<td>Long</td>
<td>Specifies the maximum number of pages to be included in one file.</td>
</tr>
<tr>
<td>WriteCharactersFormatting</td>
<td>Boolean</td>
<td>Specifies whether character formatting (e.g. font size) is to be written to files in XML format.</td>
</tr>
<tr>
<td>WriteCharAttributes</td>
<td>Boolean</td>
<td>Specifies whether character attributes (e.g. character coordinates) are to be written to files in XML format.</td>
</tr>
<tr>
<td>WriteExtendedCharAttributes</td>
<td>Boolean</td>
<td>Specifies whether extended attributes (e.g. whether a word was recognized uncertainly, whether the word was found in the dictionary) are to be written to files in XML format.</td>
</tr>
<tr>
<td>WriteNonDeskewedCoordinates</td>
<td>Boolean</td>
<td>Specifies whether character coordinates written to files in XML format are coordinates on a non-deskewed image plane.</td>
</tr>
</tbody>
</table>

See also

OutputFormatSettings
ABBY Recognition Server 4 Web Services API
TextExportSettings Object

This object provides functionality for tuning export parameters when exporting recognized text to TXT format. The TextExportSettings object is a child object of the OutputFormatSettings object and inherits all its properties.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CodePage</td>
<td>CodePageEnum</td>
<td>This property specifies the code page to which the recognized text is to be exported.</td>
</tr>
<tr>
<td>ExportParagraphsAsOneLine</td>
<td>Boolean</td>
<td>Specifies whether each paragraph in the recognized text is to be exported as one line.</td>
</tr>
<tr>
<td>InsertEmptyLineBetweenParagraphs</td>
<td>Boolean</td>
<td>Specifies if an empty line should be inserted between paragraphs.</td>
</tr>
<tr>
<td>EncodingType</td>
<td>TextEncodingTypeEnum</td>
<td>This property specifies the encoding type of the output file in TXT format. Note: If this property is set to a value different from TET_Simple, the CodePage property is ignored during export.</td>
</tr>
<tr>
<td>KeepOriginalHeadersFooters</td>
<td>Boolean</td>
<td>If this property is set to TRUE, original headers and footers will be preserved in the output file. Specifies whether page break symbols (0x12) must be inserted between pages when multiple pages are exported into TXT format.</td>
</tr>
</tbody>
</table>

See also

OutputFormatSettings
CSVExportSettings Object

This object provides functionality for tuning export parameters when exporting recognized text to CSV format. The CSVExportSettings object is a child object of the OutputFormatSettings object and inherits all its properties.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CodePage</td>
<td>CodePageEnum</td>
<td>This property specifies the code page to which the recognized text is to be exported.</td>
</tr>
<tr>
<td>EncodingType</td>
<td>TextEncodingTypeEnum</td>
<td>This property specifies the encoding type of the output file in CSV format.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Note:</strong> If this property is set to a value different from TET_Simple, the CodePage property is ignored during export.</td>
</tr>
<tr>
<td>IgnoreTextOutsideTables</td>
<td>Boolean</td>
<td>If this property is set to TRUE, only text from table blocks is exported into CSV format.</td>
</tr>
<tr>
<td>TabSeparator</td>
<td>String</td>
<td>Stores the character with which the table separators are to be replaced in the exported text. The string accessed through this property must contain only one character from the ASCII character set.</td>
</tr>
<tr>
<td>UsePageBreaks</td>
<td>Boolean</td>
<td>Specifies if page break symbols (0x12) must be inserted between pages if pages are to be exported into CSV format.</td>
</tr>
</tbody>
</table>

See also

OutputFormatSettings
PDFExportSettings Object

This object provides functionality for tuning export parameters when exporting recognized text to PDF format. The PDFExportSettings object is a child object of the OutputFormatSettings object and inherits all its properties.

When you save texts that use a non-Latin codepage (say, Cyrillic, Greek, Czech, etc.), ABBYY Recognition Server Open API uses the fonts provided by ParaType company (http://www.paratype.com/store/).

Peculiarities of exporting hieroglyphic languages to PDF

You can export hieroglyphic languages to PDF in any mode other than PDF Image Only (ExportMode = PEM_ImageOnly). For correct depicting of hieroglyphic characters, files for East Asian languages must be installed on all computers where the Processing Station component is installed.

When exporting to PDF, the following fonts are used:

- for the ChineseTraditional recognition language – the PMingLiU font and the MingLiU monospace font;
- for the ChineseSimplified recognition language – the SimSun font;
- for the Japanese recognition language – the MS PMincho font and the MS Mincho monospace font;
- for the Korean recognition language – the Batang font and the BatangChe monospace font.

Note: The earliest version of the PDF file which matches the specified properties of the PDFEncryptionInfo object is selected as the version of the PDF file.

- The earliest file version available is version 1.3.
- If at least one of the AllowFillingFormFields, AllowExtractingTextAndGraphicsExt, AllowDocumentAssembling or AllowPrintingExt properties is TRUE, or the EncryptionLevel property is PEL_High, the PDF file version will be 1.4.
- If the EncryptionLevel property is PEL_HighAES, the version will be 1.6.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ColorControlType</td>
<td>PictureColorControlTypeEnum</td>
<td>Sets a color mode for output PDF file: Auto, ColorToBwBinary, ColorToGray or SaveColoration.</td>
</tr>
<tr>
<td>Property</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>-----------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>DocumentInfo</td>
<td>Stores the metadata from the PDF file. Specifies encryption parameters of the PDF file. The property returns a reference to the PDFEncryptionInfo object.</td>
<td></td>
</tr>
<tr>
<td>EncryptionInfo</td>
<td>Specifies the parameters of the header and footer for the PDF document.</td>
<td></td>
</tr>
<tr>
<td>IsEncryptionRequested</td>
<td>Specifies whether the PDF file must be encrypted. If this property is set to TRUE, original headers and footers will be preserved in the output file.</td>
<td></td>
</tr>
<tr>
<td>KeepOriginalHeadersFooters</td>
<td>Specifies the mode of export of recognized text into PDF format. This may be: text and pictures only, text over the page image, text under the page image, page image only.</td>
<td></td>
</tr>
<tr>
<td>PDFExportMode</td>
<td>Specifies a PDF version.</td>
<td></td>
</tr>
<tr>
<td>PDFVersion</td>
<td>Specifies if document metadata (the key-value pairs of the DocumentInfo property and the values of the Title, Author, Subject, Producer, Keywords properties) must be used instead of the properties of the source document. The default value is FALSE.</td>
<td></td>
</tr>
<tr>
<td>OverwriteMetadata</td>
<td>Note: If at least one of the Title, Author, Subject, Producer, Keywords, and DocumentInfo properties is changed, this property is automatically set to TRUE.</td>
<td></td>
</tr>
</tbody>
</table>
PaperHeight: Long  
Stores paper height in twips (1/1440 of inch). See standard paper sizes in the table below. The value of this property is ignored, if the UseOriginalPaperSize property is set to TRUE.

PaperWidth: Long  
Stores paper width in twips (1/1440 of inch). See standard paper sizes in the table below. The value of this property is ignored, if the UseOriginalPaperSize property is set to TRUE.

PictureResolution: Integer  
Stores the value of picture resolution in dpi, which is used for exporting pictures into PDF format. This property may be set to -1, which means that the original resolution must be preserved.

Quality: Integer  
Stores the value in percentage points of the JPEG quality for color pictures saved in PDF format. This value is ignored for black-and-white pictures.

ReplaceUncertainWordsWithImages: Boolean  
Specifies if uncertainly recognized words will be replaced with their images when exporting into PDF format. You may use this property when the ExportMode property is set to PEM_TextAndPictures or PEM_TextOnImage, otherwise its value is ignored.

Scenario: String  
Specifies the export profile. This may be: BestQuality,
UseImprovedCompression  Boolean

If this property is TRUE, the images in the PDF file will be compressed using Mixed Raster Content (MRC) technology. MRC will provide better compression for some images than the commonly used JPEG method. For example, MRC will better compress good quality images with high-contrast text against a uniform background. Use MRC to reduce the size of the resulting PDF file. The default value is FALSE.

Note: When using MRC on low contrast images with a parti-coloured background artifacts are possible.

If this property is TRUE, original paper size is retained during export to PDF format. If the value of this property is FALSE, the paper size specified in the PaperWidth and PaperHeight properties is used.

WriteAnnotations (Obsolete)  Boolean

Specify whether to write annotations when creating a PDF file. The default value is TRUE.

Specify if the recognized text should be exported to tagged PDF. Tagged PDF is a particular use of structured PDF that allows page content to be extracted and used for various
purposes such as reflow of text and graphics, conversion to file formats such as HTML and XML, and accessibility to the visually impaired.

* – Note: If you change the values of the **Author**, **Keywords**, **Subject** or **Title** properties the values of the corresponding metadata keys of the **DocumentInfo** property will be changed or the corresponding key-value pairs will be added to the collection. Vice versa, if the values of the Author, Keywords, Subject or Title keys of the **DocumentInfo** property are set to some value, the values of the corresponding properties of the **PDFExportSettings** object will be set to the same value.

### Paper size in different units of measurement

<table>
<thead>
<tr>
<th>Paper size in inch</th>
<th>in mm</th>
<th>in twips (1/1440 of inch)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A3</td>
<td>11,69 x 16,54</td>
<td>297 x 420 16838 x 23811</td>
</tr>
<tr>
<td>A4</td>
<td>8,27 x 11,69</td>
<td>210 x 297 11909 x 16834</td>
</tr>
<tr>
<td>A5</td>
<td>5,83 x 8,27</td>
<td>148 x 210 8391 x 11909</td>
</tr>
<tr>
<td>Legal</td>
<td>8,5 x 14</td>
<td>216 x 356 12240 x 20160</td>
</tr>
<tr>
<td>Letter</td>
<td>8,5 x 11</td>
<td>216 x 279 12240 x 15840</td>
</tr>
<tr>
<td>Executive</td>
<td>7,25 x 10,5</td>
<td>184 x 266 10440 x 15120</td>
</tr>
</tbody>
</table>

See also

**OutputFormatSettings**. See sample: **WebServiceHello**.
ABBYY Recognition Server 4 Web Services API
PDFExportSettings Object

This object provides functionality for tuning export parameters when exporting recognized text to PDF/A format. The PDFExportSettings object is a child object of the OutputFormatSettings object and inherits all its properties.

When you save texts that use a non-Latin codepage (say, Cyrillic, Greek, Czech, etc.), ABBYY Recognition Server Open API uses the fonts provided by ParaType company (http://www.paratype.com/store/).

Peculiarities of exporting hieroglyphic languages to PDF/A

You can export hieroglyphic languages to PDF/A in any mode other than PDF Image Only (PDFExportMode = PEM_ImageOnly). For correct depicting of hieroglyphic characters, files for East Asian languages must be installed on all computers where the Processing Station component is installed.

When exporting to PDF/A, the following fonts are used:

- for the ChineseTraditional recognition language – the PMingLiU font and the MingLiU monospace font;
- for the ChineseSimplified recognition language – the SimSun font;
- for the Japanese recognition language – the MS PMincho font and the MS Mincho monospace font;
- for the Korean recognition language – the Batang font and the BatangChe monospace font.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ColorControlType</td>
<td>PictureColorControlTypeEnum</td>
<td>Note: These settings will be ignored if UseImprovedCompression is TRUE.</td>
</tr>
<tr>
<td>DocumentInfo</td>
<td>DocumentInfoItem Array</td>
<td>Stores the metadata from the PDF/A file.</td>
</tr>
<tr>
<td>HeaderAndFooter</td>
<td>HeaderAndFooterSettings</td>
<td>Specifies the parameters of the header and footer for the PDF/A document. If this property is set to TRUE,</td>
</tr>
</tbody>
</table>
KeepOriginalHeadersFooters  **Boolean**

Specifies if document metadata (the key-value pairs of the **DocumentInfo** property and the values of the **Title**, **Author**, **Subject**, **Producer**, **Keywords** properties*) must be used instead of the properties of the source document. The default value is FALSE.

**Note:** If at least one of the **Title**, **Author**, **Subject**, **Producer**, **Keywords**, and **DocumentInfo** properties is changed, this property is automatically set to TRUE.

---

**OverwriteMetadata  **Boolean**

Stores paper height in twips (1/1440 of inch). See the table "**Paper size in different units of measurement**". The value of this property is ignored, if the **UseOriginalPaperSize** property is set to TRUE.

---

**PaperHeight  Long**

Stores paper width in twips (1/1440 of inch). See the table "**Paper size in different units of measurement**". The value of this property is ignored, if the **UseOriginalPaperSize** property is set to TRUE.

---

**PaperWidth  Long**

Specifies the mode of export of recognized text into PDF/A format. It may be: text under the page image (PEM_ImageOnText) and page image only (PEM_ImageOnly).

---

**PDFExportMode  PDFExportModeEnum**

Specifies the PDF/A standard to be used for the output file.

---

**PDFAMode  PDFAModeEnum**

Specifies a PDF version.

---

**PDFVersion  PDFVersionEnum**

Stores the value of picture resolution in dpi, which is used for exporting pictures into PDF/A format. This property may be set to
Quality  Integer

Stores the value in percentage points of the JPEG quality for color pictures saved in PDF/A format. This value is ignored for black-and-white pictures.

Scenario  String

Specifies the export profile. This may be: BestQuality, MinSize, MaxSpeed or Balanced. The default value is Balanced.

If this property is TRUE, the images in the PDF file will be compressed using Mixed Raster Content (MRC) technology. MRC will provide better compression for some images than the commonly used JPEG method. For example, MRC will better compress good quality images with high-contrast text against a uniform background.

UseMRC to reduce the size of the resulting PDF file. The default value is FALSE.

Note: When using MRC on low contrast images with a particoloured background artifacts are possible.

If this property is TRUE, original paper size is retained during export to PDF/A format. If the value of this property is FALSE, the paper size specified in the PaperWidth and PaperHeight properties is used.

Specify whether to write annotations when creating a PDF file. The default value is TRUE.

* — Note: If you change the values of the Author, Keywords, Subject or Title properties the values of the corresponding metadata keys of the DocumentInfo property will be changed or the corresponding key-value pairs will be added to the collection. Vice versa, if the values of the Author, Keywords, Subject or Title keys of the DocumentInfo property are set to some value, the values of the corresponding properties of the PDFAEExportSettings object will be set to the same value.
See also

OutputFormatSettings
EPUBExportSettings Object

This object provides functionality for tuning export parameters when exporting recognized text to EPUB format. The EPUBExportSettings object is a child object of the OutputFormatSettings object and inherits all its properties.

When you save texts that use a non-Latin codepage (say, Cyrillic, Greek, Czech, etc.), ABBYY Recognition Server Open API uses the fonts provided by ParaType company (http://www.paratype.com/store/).

Peculiarities of exporting hieroglyphic languages to EPUB

You can export hieroglyphic languages to EPUB. For correct depicting of hieroglyphic characters, files for East Asian languages must be installed on all computers where the Processing Station component is installed.

When exporting to EPUB, the following fonts are used:

- for the ChineseTraditional recognition language – the PMingLiU font and the MingLiU monospace font;
- for the ChineseSimplified recognition language – the SimSun font;
- for the Japanese recognition language – the MS PMincho font and the MS Mincho monospace font;
- for the Korean recognition language – the Batang font and the BatangChe monospace font.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EpubVersion</td>
<td>EpubVersionEnum</td>
<td>Specifies a EPUB version.</td>
</tr>
<tr>
<td>FontFormattingMode</td>
<td>FontFormattingModeEnum</td>
<td>Specifies a layout retention method for files saved in EPUB format. The default value of this property is FAM_Restricted.</td>
</tr>
<tr>
<td>KeepPictures</td>
<td>Boolean</td>
<td>Specifies whether to retain pictures in the EPUB file. The default value is TRUE.</td>
</tr>
<tr>
<td>PictureFormat</td>
<td>ExportPictureFormatEnum</td>
<td>Stores the value of picture resolution in dpi, which is used for exporting pictures</td>
</tr>
<tr>
<td>Property</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>----------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>PictureResolution</td>
<td>Integer</td>
<td>Stores the value in percentage points of the JPEG quality for color pictures saved in PDF format. This value is ignored for black-and-white pictures.</td>
</tr>
<tr>
<td>Quality</td>
<td>Integer</td>
<td>Specifies whether the first page should be treated as a cover. The default value is TRUE.</td>
</tr>
</tbody>
</table>

See also

*OutputFormatSettings*. See sample: [WebServiceHello](#).
ABBYY Recognition Server 4 Web Services API
TiffExportSettings Object

This object provides functionality for tuning export parameters when exporting to TIFF format. The TiffExportSettings object is a child object of the OutputFormatSettings object and inherits all its properties.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ColorMode</td>
<td>ImageColorModeEnum</td>
<td>Specifies image color.</td>
</tr>
<tr>
<td>Compression</td>
<td>ImageCompressionTypeEnum</td>
<td>Specifies the compression. If the ColorMode property is set to ICM_AsIs, ICM_Color, or ICM_Gray, this property can have one of the following values: ICT_Uncompressed, ICT_Zip, ICT_Jpeg. If the ColorMode property is set to ICM_BlackAndWhite, this property can have one of the following values: ICT_Uncompressed, ICT_Zip, ICT_Group3, ICT_Group4, ICT_PackBits.</td>
</tr>
<tr>
<td>Resolution</td>
<td>Integer</td>
<td>Specifies the value of picture resolution in dpi, which is used for exporting. The value &quot;-1&quot; means that the original resolution must be preserved.</td>
</tr>
</tbody>
</table>

See also

OutputFormatSettings
ABBYY Recognition Server 4 Web Services API
Jpeg2kExportSettings Object

This object provides functionality for tuning export parameters when exporting to JPEG 2000 format. The Jpeg2kExportSettings object is a child object of the OutputFormatSettings object and inherits all its properties.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ColorMode</td>
<td>ImageColorModeEnum</td>
<td>Specifies image color. This property can be set to one of the following values: ICM_AsIs, ICM_Color, ICM_Gray.</td>
</tr>
<tr>
<td>Quality</td>
<td>Integer</td>
<td>Specifies the quality in percentage points.</td>
</tr>
<tr>
<td>Resolution</td>
<td>Integer</td>
<td>Stores the value of picture resolution in dpi, which is used for exporting. The value &quot;-1&quot; means that the original resolution must be preserved.</td>
</tr>
</tbody>
</table>

See also

OutputFormatSettings
ABBYY Recognition Server 4 Web Services API
This object provides functionality for tuning export parameters when exporting to JPEG format. The JpegExportSettings object is a child object of the OutputFormatSettings object and inherits all its properties.

### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ColorMode</td>
<td>ImageColorModeEnum</td>
<td>Specifies image color. This property can be set to one of the following values: ICM_AsIs, ICM_Color, ICM_Gray.</td>
</tr>
<tr>
<td>Quality</td>
<td>Integer</td>
<td>Specifies the quality in percentage points.</td>
</tr>
<tr>
<td>Resolution</td>
<td>Integer</td>
<td>Stores the value of picture resolution in dpi, which is used for exporting. The value &quot;-1&quot; means that the original resolution must be preserved.</td>
</tr>
</tbody>
</table>

See also

OutputFormatSettings
JBig2ExportSettings Object

This object provides functionality for tuning export parameters when exporting to JBIG2 format. The JBig2ExportSettings object is a child object of the OutputFormatSettings object and inherits all its properties.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resolution</td>
<td>Integer</td>
<td>Specifies the value of picture resolution in dpi, which is used for exporting. The value &quot;-1&quot; means that the original resolution must be preserved.</td>
</tr>
</tbody>
</table>

See also

OutputFormatSettings
AltoExportSettings Object
(IAaltoExportSettings Interface)

This object provides functionality for tuning export parameters when exporting recognized text to Alto XML format. The IAltoExportSettings interface is a child object of the IOutputFormatSettings interface and inherits all its properties. It is an element of the OutputFormatSettingsCollection collection.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TextCoordinatesParticularity</td>
<td>TextCoordinatesParticularityEnum</td>
<td>Specifies how text should be divided: by words or by lines. This property can be set to one of the following values: TCP_Words, TCP_Lines. The default value is TCP_Words.</td>
</tr>
<tr>
<td>FontFormattingMode</td>
<td>FontFormattingModeEnum</td>
<td>Select the desired font formatting mode: plain, restricted or full. This property can be set to one of the following values: FFM_Full, FFM_Plain, FFM_Restricted. The default value is FFM_Plain.</td>
</tr>
<tr>
<td>MeasurementUnit</td>
<td>AltoMeasurementUnitEnum</td>
<td>Specifies the measurement unit used to describe size and coordinates of objects in the output XML file. This property can be set to one of the following values: AMU_Inch1200, AMU_Mm10, AMU_Pixel. The default value is AMU_Pixel.</td>
</tr>
</tbody>
</table>

See also

OutputFormatSettings, OutputFormatSettingsCollection
PDFEncryptionInfo Object

This object provides access to encryption parameters of the PDF file during export. These parameters are set in the `EncryptionInfo` property of `PDFExportSettings`. The PDFEncryptionInfo object allows you to do the following:

- set owner and user passwords;
- set the level of encryption;
- enable or disable the following:
  - adding or modifying text annotations and interactive form fields;
  - assembling the document: inserting, rotating, or deleting pages and creating navigation elements such as bookmarks or thumbnail images;
  - copying or otherwise extracting text and graphics from the document;
  - filling out forms (that is, filling out existing interactive form fields) and signing the document (which amounts to filling out existing signature fields, a type of interactive form field);
  - modifying the contents of the document;
  - printing the document.

**Note:** The earliest version of the PDF file which matches the specified properties of the PDFEncryptionInfo object is selected as the version of the PDF file.

- The earliest file version available is version 1.3.
- If at least one of the `AllowFillingFormFields`, `AllowExtractingTextAndGraphicsExt`, `AllowDocumentAssembling` or `AllowPrintingExt` properties is TRUE, or the `EncryptionLevel` property is PEL_High, the PDF file version will be 1.4.
- If the `EncryptionLevel` property is PEL_HighAES, the version will be 1.6.

### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AllowAddingTextAnnotations</td>
<td>Boolean</td>
<td>Enables/disables adding or modifying text annotations and interactive form fields.</td>
</tr>
<tr>
<td>AllowDocumentAssembling</td>
<td>Boolean</td>
<td>Enables/disables assembling the document: inserting, rotating, or deleting pages and creating navigation elements such as bookmarks or thumbnail images.</td>
</tr>
<tr>
<td>AllowExtractingTextAndGraphics</td>
<td>Boolean</td>
<td>Enables/disables copying or otherwise extracting text and graphics from the document.</td>
</tr>
<tr>
<td>Property</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------------</td>
<td>-----------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>AllowExtractingTextAndGraphicsExt</td>
<td>Boolean</td>
<td>Enables/disables extracting text and graphics (to make the accessible to users with disabilities or for other purposes).</td>
</tr>
<tr>
<td>AllowFillingFormFields</td>
<td>Boolean</td>
<td>Enables/disables filling out forms (that is, filling out existing interactive form fields) and signing the document (which amounts to filling out existing signature fields, a type of interactive form field).</td>
</tr>
<tr>
<td>AllowModifyingContent</td>
<td>Boolean</td>
<td>Enables/disables modifying the contents of the document.</td>
</tr>
<tr>
<td>AllowPrinting</td>
<td>Boolean</td>
<td>Enables/disables printing the document.</td>
</tr>
<tr>
<td>AllowPrintingExt</td>
<td>Boolean</td>
<td>Enables/disables printing to a representation from which a faithful digital copy of the PDF content could be generated. Disallowing such printing may result in degradation of output quality (a feature implemented as &quot;Print As Image&quot; in Acrobat).</td>
</tr>
<tr>
<td>EncryptionLevel</td>
<td>PDFEncryptionLevelEnum</td>
<td>Sets the level of encryption.</td>
</tr>
<tr>
<td>OwnerPassword</td>
<td>String</td>
<td>Stores the owner password. Opening the document with the correct owner password (assuming it is not the same as the user password) allows full (owner) access to the document. This unlimited access includes the ability to change the document’s passwords and access permissions.</td>
</tr>
<tr>
<td>UserPassword</td>
<td>String</td>
<td>Stores the user password. Opening the document with the correct user password (or opening a document that does not have a user password) allows additional operations to be performed according to the</td>
</tr>
</tbody>
</table>
See also

PDFExportSettings::EncryptionInfo
StringsCollection Object (IStringsCollection Interface)

This object provides access to a collection of strings.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count</td>
<td>Long, read-only</td>
<td>Stores the number of elements in the collection.</td>
</tr>
</tbody>
</table>

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add</td>
<td>Adds a new element at the end of the collection.</td>
</tr>
<tr>
<td>Item</td>
<td>Provides access to a single element of the collection.</td>
</tr>
<tr>
<td>Remove</td>
<td>Removes an element from the collection.</td>
</tr>
<tr>
<td>RemoveAll</td>
<td>Removes all the elements from the collection.</td>
</tr>
</tbody>
</table>

See also

IClient::Workflows, IInputFile::Errors, IInputFile::Warnings, IXmlResult::Errors, IXmlResult::Warnings, IOutputDocument::FileNames, IRecognitionParams::Languages, Working with Collections.

See samples: Hello, Listening, AsyncProcessing, Sample for ASP.NET.
ABBYY Recognition Server 4 COM-based API
Statistics Object (IStatistics Interface)

This object represents statistics about the results of recognition.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PagesArea</td>
<td>Integer, read-only</td>
<td>Returns the total area of recognized pages measured in A4 pages. This is the number that was used to increment the page counter for the license.</td>
</tr>
<tr>
<td>TotalCharacters</td>
<td>Integer, read-only</td>
<td>Returns the total number of recognized characters.</td>
</tr>
<tr>
<td>UncertainCharacters</td>
<td>Integer, read-only</td>
<td>Returns the total number of uncertain characters.</td>
</tr>
</tbody>
</table>

See also

IXmlResult::Statistics
HeaderAndFooterSettings Object

This object provides functionality for tuning header and footer parameters when exporting recognized text to PDF (PDF/A) format. These parameters are set in the HeaderAndFooter property of the PDFExportSettings (PDFAExportSettings) object. The HeaderAndFooterSettings object allows you to do the following:

- Place headers and footers on each document page;
- Specify the text to be stamped, the location of the header or footer on the page, and other formatting options;
- Include into the header or footer any static text, as well as page numbers, Bates numbers (auto-incrementing number), current date and time, output file name, etc. To include a variable into the footer or header text, insert a suitable tag from the list below:
  - <BatesNum> Auto-incrementing numbers
  - <Dd> Current date
  - <EmailSubject> Subject of the incoming e-mail message
  - <FileName> Name of the output file
  - <Folder> Name of parent subfolder in image folder
  - <Mm> Current month
  - <PageNum> Page number
  - <Yy> Current year (2 digits)
  - <Yyyy> Current year (4 digits)

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BottomMargin</td>
<td>Double</td>
<td>Sets the bottom margin. The value of this property must be in the range of 0 to 120 inches (0 to 3048 millimeters).</td>
</tr>
<tr>
<td>CentralFooter</td>
<td>String</td>
<td>Specifies the central footer text.</td>
</tr>
<tr>
<td>CentralHeader</td>
<td>String</td>
<td>Specifies the central header text.</td>
</tr>
<tr>
<td>FontName</td>
<td>String</td>
<td>Sets the font name.</td>
</tr>
<tr>
<td>FontSize</td>
<td>Long</td>
<td>Sets the font size in points.</td>
</tr>
<tr>
<td>IsBold</td>
<td>Boolean</td>
<td>Specifies if the bold font style has to be used.</td>
</tr>
</tbody>
</table>
IsInInches  **Boolean**  Specifies whether measurement units for the margins are inches. If the value of this property is FALSE, the measurement units for the margins are millimeters.

IsItalic  **Boolean**  Specifies if the italic font style has to be used.

IsUnderlined  **Boolean**  Specifies if the underlined font style has to be used.

LeftFooter  **String**  Specifies the left footer text.

LeftHeader  **String**  Specifies the left header text.

LeftMargin  **Double**  Sets the left margin. The value of this property must be in the range of 0 to 120 inches (0 to 3048 millimeters).

NumberOfDigits  **Long**  Sets the number of digits in the Bates numbers. If the number of digits in a Bates number is less than specified by this property, the corresponding number of zeros will be added in front of the Bates number. The maximal allowed number of digits is 9.

RightFooter  **String**  Specifies the right footer text.

RightHeader  **String**  Specifies the right header text.

RightMargin  **Double**  Sets the right margin. The value of this property must be in the range of 0 to 120 inches (0 to 3048 millimeters).

StartingNumber  **Long**  Sets the starting Bates number. The number of digits in the starting number must not exceed the number specified in the **NumberOfDigits** property.

TextColor  **Long**  Sets the text color in RGB format.

**Note:** The Long value is calculated from the RGB triplet using the formula: $(\text{red value}) + (256 \times \text{green value}) + (65536 \times \text{blue value})$, where \text{red value} is the first triplet component, \text{green value} is the second triplet component, \text{blue value} is the third triplet component.

TopMargin  **Double**  Sets the top margin. The value of this property must be in the range of 0 to 120 inches (0 to 3048 millimeters).

See also

**PDFExportSettings::HeaderAndFooter**,  **PDFAExportSettings::HeaderAndFooter**
DocumentInfoItem Object

This object contains metadata from the PDF (PDF/A) file. It represents a key-value pair.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>String</td>
<td>Stores the name of metadata which is used as a key.</td>
</tr>
<tr>
<td>Value</td>
<td>String</td>
<td>Stores the value of metadata.</td>
</tr>
</tbody>
</table>

See also

PDFExportSettings::DocumentInfoItem, PDFAExportSettings::DocumentInfoItem
ABBY Recognition Server 4 Web Services API
AltoMeasurementUnitEnum enumeration constants are used to set the measurement unit used to describe size and coordinates of objects in the output XML file.

```java
enum AltoMeasurementUnitEnum{
    AMU_Inch1200,
    AMU_Mm10,
    AMU_Pixel
}
```

Elements

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AMU_Inch1200</td>
<td>The unit of measure is inch/1200.</td>
</tr>
<tr>
<td>AMU_Mm10</td>
<td>The unit of measure is mm/10.</td>
</tr>
<tr>
<td>AMU_Pixel</td>
<td>The unit of measure is pixel.</td>
</tr>
</tbody>
</table>

See also

AltoExportSettings
ABBYY Recognition Server 4 Web Services  API
**AttributeTypeEnum** enumeration constants define different types of document attributes.

```c
typedef enum {
    AT_Boolean,
    ATEnumeration,
    ATSingleLine,
    ATMultipleLines
    ATRegularExpression
} AttributeTypeEnum;
```

**Elements**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AT_Boolean</td>
<td>Checkbox.</td>
</tr>
<tr>
<td>ATEnumeration</td>
<td>Element of the list.</td>
</tr>
<tr>
<td>ATSingleLine</td>
<td>Single line.</td>
</tr>
<tr>
<td>ATMultipleLines</td>
<td>Multiple lines.</td>
</tr>
<tr>
<td>ATRegularExpression</td>
<td>Regular expression.</td>
</tr>
</tbody>
</table>

**See also**

[DocumentAttribute](#)
ABBYY Recognition Server 4 Web Services API
BarcodeTypeEnum

BarcodeTypeEnum enumeration constants are used to specify the type of barcode used for separating documents.

```java
BarcodeTypeEnum enum {
    IBT_Code39,
    IBT_CheckCode39,
    IBT_Code39WithoutAsterisk,
    IBT_Interleaved25,
    IBT_CheckInterleaved25,
    IBT_Ean13,
    IBT_Code128,
    IBT_Ean8,
    IBT_Pdf417,
    IBT_Codabar,
    IBT_Upce,
    IBT_Industrial25,
    IBT_Iata25,
    IBT_Matrix25,
    IBT_Code93,
    IBT_Postnet,
    IBT_Ucc128,
    IBT_Patch,
    IBT_Upca,
    IBT_Aztec,
    IBT_Datamatrix,
    IBT_Qrcode,
    IBT_MaxiCode,
    IBT_Code32,
    IBT_FullASCII,
    IBT_Royal,
    IBT_Kix,
    IBT_Intelligent,
    IBT_Unknown
};
```

Elements

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBT_Aztec</td>
<td>Aztec is a high density two-dimensional matrix style barcode that can encode up to 3750 characters from the entire 256 byte ASCII character set. The symbol is built on a square grid with a bulls-eye pattern at its center.</td>
</tr>
<tr>
<td>IBT_CheckCode39</td>
<td>Check Code 39 is a Code 39 barcode with a check sum.</td>
</tr>
<tr>
<td>IBT_CheckInterleaved25</td>
<td>Check Interleaved 2 of 5 is an Interleaved 2 of 5 barcode with a check sum.</td>
</tr>
<tr>
<td>IBT_Codabar</td>
<td>Codabar is a self-checking, variable length barcode that can encode 16 data characters. It is used primarily for numeric data, but also</td>
</tr>
</tbody>
</table>
IBT_Codabar encodes six special characters. Codabar is useful for encoding dollar and mathematical figures because a decimal point, plus sign, and minus sign can be encoded.

Code 128 is an alphanumeric, very high-density, compact, variable length barcode scheme that can encode the full 128 ASCII character set. Each character is represented by three bars and three spaces totaling 11 modules. Each bar or space is one, two, three, or four modules wide with the total number of modules representing bars an even number and the total number of modules representing a space an odd number. Three different start characters are used to select one of three character sets.

Code 39, also referred to as Code 3 of 9, is an alphanumeric, self-checking, variable length barcode that uses five black bars and four spaces to define a character. Three of the elements are wide and six are narrow.

Code 39 without asterisk is a Code 39 barcode, which has no start and stop symbol, the asterisk "*".

Note: In ABBYY Recognition Serve, barcodes of type Code 39 without asterisk can only be used for document separation, as they are recognized only if the user has explicitly specified their type.

Code 93 is a variable length bar code that encodes 47 characters. It is named Code 93 because every character is constructed from nine elements arranged into three bars with their adjacent spaces. Code 93 is a compressed version of Code 39 and was designed to complement Code 39.

Data Matrix is a two-dimensional matrix barcode consisting of black and white modules arranged in either a square or rectangular pattern. Every Data Matrix is composed of two solid adjacent borders in an "L" shape and two other borders consisting of alternating dark and light modules. Within these borders are rows and columns of cells encoding information. A Data Matrix barcode can store up to 2335 alphanumeric characters.

The European Article Numbering (EAN) system is used for products that require a country origin. This is a fixed-length barcode used to encode either eight or thirteen characters. The first two characters identify the country of origin, the next characters are data characters, and the last character is the checksum.

IATA 2 of 5 is a numeric, variable length barcode. It is a barcode standard designed by the IATA (International Air Transport Association). This standard is used for all boarding passes.

Industrial 2 of 5 is numeric-only barcode that has been in use a long time. Unlike Interleaved 2 of 5, all of the information is encoded in the bars; the spaces are fixed width and are used only to separate the
Intelligent Mail Barcode (IM barcode) is a barcode that encodes up to 31 digits of information about the route and delivery means of a mail piece. It contains 65 vertical bars of four types, each of which consists of a central portion (a bar that contains this portion only is also referred to as a tracker), and may include a top portion (ascender), a bottom portion (descender) or both (full bar). The 65 bars represent ten 13-bit characters, totaling 130 bits. Each of these characters contains 2, 5, 8 or 11 bits set to one, and the Hamming distance between characters is at least 2. Due to this, single-bit errors in characters can be automatically detected and corrected. The Intelligent Mail Barcode has been adopted by the United States Postal Services.

Interleaved 2 of 5 is a variable length (must be a multiple of two), high-density, self-checking, numeric barcode that uses five black bars and five white bars to define a character. Two digits are encoded in every character; one in the black bars and one in the white bars. Two of the black bars and two of the white bars are wide. The other bars are narrow.

Standard 2 of 5 is self-checking numeric-only barcode. Unlike Interleaved 2 of 5, all of the information is encoded in the bars; the spaces are fixed width and are used only to separate the bars. Matrix 2 of 5 is used primarily for warehouse sorting, photo finishing, and airline ticket marking.

Patch Code is 1 character long barcode, which does not encode data, but acts as a signal. It is used only for batch separation and scanner control.

PDF417 is a variable length, two-dimensional (2D), stacked symbology that can store up to 1850 printable ASCII characters or 1100 binary characters per symbol. PDF417 is designed with selectable levels of error correction. Its high data capacity can be helpful in applications where a large amount of data must travel with a labeled document or item.

The Postnet (Postal Numeric Encoding Technique) is a fixed length symbology (5, 6, 9, or 11 characters) which uses constant bar and space width. Information is encoded by varying the bar height between the two values. Postnet barcodes are placed on the lower right of envelopes or postcards, and are used to expedite the processing of mail with automatic equipment and provide reduced postage rates.

QR Code is a two-dimensional matrix barcode. The barcode has 3 large squares (registration marks) in the corners which define the top of the barcode. The black and white squares in the area between the registration marks are the encoded data and error correction keys.
QR Codes can encode over 4000 ASCII characters.

This type of barcode is a 19 digit barcode with a 20th check digit. For a total of 20 digits. It typically is used for carton identification. Both for internal carton numbering and also for using the UCC-128 barcode on your cartons being shipped out to your customers.

If this value is specified, the program will use barcodes of all types to separate documents.

The UPC-A (Universal Product Code) barcode is 12 digits long, including its checksum. Each digit is represented by a seven-bit sequence, encoded by a series of alternating bars and spaces. UPC-A is used for marking products which are sold at retail in the USA. Note that UPC-A codes with 2 or 5 digit supplemental codes appended to them are not supported.

The UPC-E barcode is a shortened version of UPC-A barcode. It compresses the data characters and the checksum into six characters. This bar code is ideal for small packages because it is the smallest bar code. Note that UPC-E codes with 2 or 5 digit supplemental codes appended to them are not supported.

See also

IExportParams::SeparationMethod
CodePageEnum

CodePageEnum enumeration is the Open API internal representation of code pages.

enum CodePageEnum{
    CP_Null,
    CP_Latin,
    CP_Cyrillic,
    CP_EasternEuropean,
    CP_Baltic,
    CP_Turkish,
    CP_US_MSDOS,
    CP_LatinI_MSDOS,
    CP_Russian_MSDOS,
    CP_Baltic_MSDOS,
    CP_Turkish_IBM,
    CP_Slavic_MSDOS,
    CP_Greek,
    CP_Greek_737,
    CP_Greek_869,
    CP_Latin_ISO,
    CP_EasternEuropean_ISO,
    CP_Turkish_ISO,
    CP_Baltic_ISO,
    CP_Cyrillic_ISO,
    CP_Greek_ISO,
    CP_KOI8,
    CP_Tatar,
    CP_Tatar_MSDOS,
    CP_Roman_Macintosh,
    CP_Greek_Macintosh,
    CP_Cyrillic_Macintosh,
    CP_Ukrainian_Macintosh,
    CP_Latin2_Macintosh,
    CP_Icelandic_Macintosh,
    CP_Turkish_Macintosh,
    CP_Croatian_Macintosh,
    CP_Armenian,
    CP_Armenian_MSDOS,
    CP_Armenian_Macintosh,
    CP_Latin5_ISO,
    CP_Cyrillic_MSDOS,
    CP_Bashkir
};

Elements

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CP_Null</td>
<td>Invalid code page.</td>
</tr>
<tr>
<td>CP_Latin</td>
<td>Windows Western Europe (1252)</td>
</tr>
<tr>
<td>CP_Cyrillic</td>
<td>Windows Cyrillic (1251)</td>
</tr>
</tbody>
</table>
See also
CSVExportSettings::CodePage,
HTMLExportSettings::CodePage,
TextExportSettings::CodePage
**DocumentSeparationMethodEnum** enumeration constants are used to set the document separation method during export.

```cpp
enum DocumentSeparationMethodEnum{
    DSM_OneFilePerImage,
    DSM_MergeIntoSingleFile
};
```

**Elements**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSM_OneFilePerImage</td>
<td>The output file will have the same number of pages as the input file.</td>
</tr>
<tr>
<td>DSM_MergeIntoSingleFile</td>
<td>The files will be merged into one document.</td>
</tr>
</tbody>
</table>

**See also**

ExportParams::SeparationMethod
ABBYY Recognition Server 4 Web Services API
EPUBVersionEnum enumeration constants specify different EPUB versions.

```cpp
enum EPUBVersionEnum{
    EV_2_0_1,
    EV_3
};
```

### Elements

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EV_2_0_1</td>
<td>EPUB version 2.0.1</td>
</tr>
<tr>
<td>EV_3</td>
<td>EPUB version 3.0</td>
</tr>
</tbody>
</table>

### See also

[EPUBExportSettings::EPUBVersion](#)
ABBYY Recognition Server 4 Web Services API
**ExportPictureFormatEnum**

`ExportPictureFormatEnum` enumeration constants are used to set the color mode and compression method for images inside the PDF (PDF/A) file.

```java
enum ExportPictureFormatEnum{
    EPF_Automatic,
    EPF_Ccitt4,
    EPF_JpegColor,
    EPF_JpegGray,
    EPF_LzwColor,
    EPF_LzwGray,
    EPF_ZipColor,
    EPF_ZipGray
};
```

**Elements**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EPF_Automatic</td>
<td>The compression method and color mode are selected automatically based on two image properties: Color Type (black and white, grayscale, or color) and Color Variety (low or high). Black-and-white images are always saved using the CCITT4 compression algorithm. Grayscale and color images are saved using the ZIP compression algorithm in the case of low color variety, and using the JPEG compression algorithm in the case of high color variety.</td>
</tr>
<tr>
<td>EPF_Ccitt4</td>
<td>This format allows you to save pictures in black-and-white mode. The CCITT4 compression algorithm is used.</td>
</tr>
<tr>
<td>EPF_JpegColor</td>
<td>Images are saved in color JPEG format. This format is suitable for documents containing color scanned or digital photos.</td>
</tr>
<tr>
<td>EPF_JpegGray</td>
<td>Images are saved in gray JPEG format. This format is suitable for scanned or digital photos saved in gray-scale mode.</td>
</tr>
<tr>
<td>EPF_LzwColor</td>
<td>Images are saved in color LZW format. This format is suitable for graphics. This format is not available for PDF/A files.</td>
</tr>
<tr>
<td>EPF_LzwGray</td>
<td>Images are saved in gray LZW format. This format is suitable for graphics and gray images. This format is not available for PDF/A files.</td>
</tr>
<tr>
<td>EPF_ZipColor</td>
<td>Images are saved in color ZIP format. This format is suitable for pictures created with paint programs.</td>
</tr>
<tr>
<td>EPF_ZipGray</td>
<td>Images are saved in gray ZIP format. This format allows you to save pictures in gray-scale mode.</td>
</tr>
<tr>
<td>EPF_J2KColor</td>
<td>Images are saved in color JPEG 2000 format. This format is suitable for documents containing color scanned or digital photos. This format is not available for PDF/A files.</td>
</tr>
<tr>
<td>EPF_J2KGray</td>
<td>Images are saved in gray JPEG 2000 format. This format is suitable for scanned or digital photos saved in gray-scale mode. This format is not available for PDF/A files.</td>
</tr>
</tbody>
</table>
Images are saved in gray JBIG2 format. This format is suitable for binary images.

See also

PDFExportSettings::PictureFormat,
PDFAEExportSettings::PictureFormat
ABBYY Recognition Server 4 Web Services API
FontFormattingModeEnum enumeration constants are used to select the desired font formatting mode: plain, restricted or full.

```java
enum FontFormattingModeEnum{
    FFM_Full,
    FFM_Plain,
    FFM_Restricted
};
```

Elements

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FFM_Full</td>
<td>The character formatting mode is <strong>Full</strong>. Produced document maintains the formatting of the original.</td>
</tr>
<tr>
<td>FFM_Plain</td>
<td>The character formatting mode is <strong>Plain</strong>. Text formatting is not preserved.</td>
</tr>
<tr>
<td></td>
<td>The character formatting mode is <strong>Restricted</strong>. Retains fonts, font sizes, and paragraphs, but does not retain the exact locations of the objects on the page or the spacing. The resulting text will be left-aligned.</td>
</tr>
</tbody>
</table>

See also

**AltoExportSettings**
**HTMLSynthesisModeEnum**

`HTMLSynthesisModeEnum` enumeration constants are used to define available modes of synthesizing of HTML code from the recognized text.

```cpp
enum HTMLSynthesisModeEnum{
    HSM_PlainText,
    HSM_FormattedStream,
    HSM_PageLayout
};
```

**Elements**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HSM_PlainText</td>
<td>Only paragraphs are retained in the recognized text by using of the <code>&lt;p&gt;</code> tag.</td>
</tr>
<tr>
<td>HSM_FormattedStream</td>
<td>Paragraphs and fonts of the recognized text are retained in the output HTML file. The <code>&lt;p&gt;</code> tag is used.</td>
</tr>
<tr>
<td>HSM_PageLayout</td>
<td>Full layout of the input page is retained by using a table.</td>
</tr>
</tbody>
</table>

**See also**

`HTMLExportSettings::HTMLSynthesisMode`
ImageColorModeEnum

ImageColorModeEnum enumeration constants are used to set the image color.

```cpp
enum ImageColorModeEnum{
    ICM_AsIs,
    ICM_Color,
    ICM_Gray,
    ICM_BlackAndWhite
};
```

Elements

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ICM_AsIs</td>
<td>Image will be exported as is.</td>
</tr>
<tr>
<td>ICM_Color</td>
<td>Color image will be exported.</td>
</tr>
<tr>
<td>ICM_Gray</td>
<td>Gray image will be exported.</td>
</tr>
<tr>
<td>ICM_BlackAndWhite</td>
<td>Black and white image will be exported.</td>
</tr>
</tbody>
</table>

See also

TiffExportSettings::ColorMode, JpegExportSettings::ColorMode, Jpeg2kExportSettings::ColorMode
ABBYY Recognition Server 4 Web Services API
ImageCompressionTypeEnum

ImageCompressionTypeEnum enumeration constants are used to set the compression files.

```cpp
enum ImageCompressionTypeEnum{
    ICT_Uncompressed,
    ICT_Zip,
    ICT_Group3,
    ICT_Group4,
    ICT_PackBits,
    ICT_Jpeg,
    ICT_Lzw
};
```

Elements

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ICT_Uncompressed</td>
<td>Image will be exported without compression.</td>
</tr>
<tr>
<td>ICT_Zip</td>
<td>Image will be exported using the ZIP compression algorithm.</td>
</tr>
<tr>
<td>ICT_Group3</td>
<td>Image will be exported using the GROUP3 compression algorithm.</td>
</tr>
<tr>
<td>ICT_Group4</td>
<td>Image will be exported using the GROUP4 compression algorithm.</td>
</tr>
<tr>
<td>ICT_PackBits</td>
<td>Image will be exported using the PACKBITS compression algorithm.</td>
</tr>
<tr>
<td>ICT_Jpeg</td>
<td>Image will be exported using the JPEG compression algorithm.</td>
</tr>
<tr>
<td>ICT_Lzw</td>
<td>Image will be exported using the LZW compression algorithm.</td>
</tr>
</tbody>
</table>

See also

TiffExportSettings::Compression
ABBYY Recognition Server 4 Web Services API
OutputFileFormatEnum

OutputFileFormatEnum enumeration constants define different file formats in which ABBYY Recognition Server can save.

```java
enum OutputFileFormatEnum {
    OFF_MSWord,
    OFF_MSExcel,
    OFF_RTF,
    OFF_XML,
    OFF_PDF,
    OFF_PDFA,
    OFF_Text,
    OFF_CSV,
    OFF_HTML,
    OFF_NoConversion,
    OFF_TIFF,
    OFF_JPG,
    OFF_J2K,
    OFF_InternalFormat,
    OFF_DOCX,
    OFF_XLSX,
    OFF_JBIG2,
    OFF_AltoXML
};
```

Elements

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OFF_MSWord</td>
<td>Microsoft Word format. The parameters of a file in this format are tuned through the MSWordExportSettings object.</td>
</tr>
<tr>
<td>OFF_MSExcel</td>
<td>Microsoft Excel format. The parameters of a file in this format are tuned through the XLExportSettings object.</td>
</tr>
<tr>
<td>OFF_RTF</td>
<td>Rich text format. The parameters of a file in this format are tuned through the RTFExportSettings object.</td>
</tr>
<tr>
<td>OFF_XML</td>
<td>XML format. The parameters of a file in this format are tuned through the XMLExportSettings object.</td>
</tr>
<tr>
<td>OFF_PDF</td>
<td>PDF format. The parameters of a file in this format are tuned through the PDFExportSettings object.</td>
</tr>
<tr>
<td>OFF_PDFA</td>
<td>PDF/A format. The parameters of a file in this format are tuned through the PDFAExportSettings object.</td>
</tr>
<tr>
<td>OFF_Text</td>
<td>Text format. The parameters of a file in this format are tuned through the TextExportSettings object.</td>
</tr>
<tr>
<td>OFF_CSV</td>
<td>CSV format. The parameters of a file in this format are tuned through the CSVExportSettings object.</td>
</tr>
<tr>
<td></td>
<td>HTML format. The parameters of a file in this format are tuned through the HTMLExportSettings object.</td>
</tr>
</tbody>
</table>
OFF_HTML  **HTMLExportSettings** object.

OFF_NoConversion Saves input file without conversion.

OFF_TIFF  TIFF format. The parameters of a file in this format are tuned through the **TiffExportSettings** object.

OFF_JPG  JPEG format. The parameters of a file in this format are tuned through the **JpegExportSettings** object.

OFF_J2K  JPEG 2000 format. The parameters of a file in this format are tuned through the **Jpeg2kExportSettings** object.

OFF_InternalFormat  **ABBYY FineReader internal format**.

OFF_DOCX  Microsoft Word 2007 format. The parameters of a file in this format are tuned through the **DOCXExportSettings** object.

OFF_XLSX  Microsoft Excel 2007 format. The parameters of a file in this format are tuned through the **XLSXExportSettings** object.

OFF_JBIG2  JBIG2 format. The parameters of a file in this format are tuned through the **JBig2ExportSettings** object.

OFF_AltoXML  AltoXML format. The parameters of a file in this format are tuned through the **AltoExportSettings** object.

See also

**OutputFormatSettings**
ABBYY Recognition Server 4 Web Services API
PDFEncryptionLevelEnum enumeration constants are used to set the level of encryption of the PDF file during export.

```cpp
enum PDFEncryptionLevelEnum{
    PEL_Low,
    PEL_High,
    PEL_HighAES
};
```

### Elements

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PEL_Low</td>
<td>Sets a low (40-bit) encryption level.</td>
</tr>
<tr>
<td>PEL_High</td>
<td>Sets a high (128-bit) encryption level, but Acrobat 3.0 users cannot open PDF documents with this encryption level.</td>
</tr>
<tr>
<td>PEL_HighAES</td>
<td>Sets a high (128-bit AES) encryption level, but Acrobat 6.0 (or earlier) users cannot open PDF documents with this encryption level.</td>
</tr>
</tbody>
</table>

See also

[PDFEncryptionInfo::EncryptionLevel]
ABBYY Recognition Server 4 Web Services API
**PDFExportModeEnum**

**PDFExportModeEnum** enumeration constants are used to set the mode of export into PDF format.

```cpp
enum PDFExportModeEnum{
    PEM_TextAndPictures,
    PEM_TextOnImage,
    PEM_ImageOnText,
    PEM_ImageOnly
};
```

**Elements**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PEM_TextAndPictures</td>
<td>The recognized text is saved as text, and the pictures are saved as pictures. The original document design (font, background, and layout marking) is not retained.</td>
</tr>
<tr>
<td>PEM_TextOnImage</td>
<td>The entire image is saved as a picture. Text areas are saved as text over the image.</td>
</tr>
<tr>
<td>PEM_ImageOnText</td>
<td>The entire image is saved as a picture. The recognized text is put under it. This option is useful if you export your text to document archives: the full page layout is retained and the full-text search is available if you save in this mode.</td>
</tr>
<tr>
<td>PEM_ImageOnly</td>
<td>The entire image is saved as a picture. The recognized text and layout information are not used in this mode, so the recognition stage may be skipped.</td>
</tr>
</tbody>
</table>

**See also**

`PDFExportSettings::PDFExportMode`
PDFVersionEnum

PDFVersionEnum enumeration constants specify different PDF versions.

```cpp
enum PDFVersionEnum{
    PVN_Auto,
    PVN_Version13,
    PVN_Version14,
    PVN_Version15,
    PVN_Version16,
    PVN_Version17
};
```

Elements

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PVN_Auto</td>
<td>The version is detected automatically.</td>
</tr>
<tr>
<td>PVN_Version13</td>
<td>PDF version 1.3.</td>
</tr>
<tr>
<td>PVN_Version14</td>
<td>PDF version 1.4.</td>
</tr>
<tr>
<td>PVN_Version15</td>
<td>PDF version 1.5.</td>
</tr>
<tr>
<td>PVN_Version16</td>
<td>PDF version 1.6.</td>
</tr>
<tr>
<td>PVN_Version17</td>
<td>PDF version 1.7.</td>
</tr>
</tbody>
</table>

See also

PDFExportSettings::PDFVersion, PDFAExportSettings::PDFVersion
PictureColorControlTypeEnum enumeration constants are used to set color mode for output PDF file.

```java
gen enum ImageColorModeEnum{
    PCCT_Auto,
    PCCT_ColorToBwBinary,
    PCCT_ColorToGray,
    PCCT_SaveColoration
};
```

Elements

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCCT_Auto</td>
<td>Automatic.</td>
</tr>
<tr>
<td>PCCT_ColorToBwBinary</td>
<td>Convert color and grayscale images to black-and-white using binarization.</td>
</tr>
<tr>
<td>PCCT_ColorToGray</td>
<td>Convert color images to grayscale.</td>
</tr>
<tr>
<td>PCCT_SaveColoration</td>
<td>Keep original color mode.</td>
</tr>
</tbody>
</table>

See also

[PDFExportSettings](#), [PDFAExportSettings](#)
ABBYY Recognition Server 4 Web Services API
PriorityEnum enumeration constants are used to set priorities of the jobs.

```cpp
enum PriorityEnum{
    P_Low,
    P_BelowNormal,
    P_Normal,
    P_AboveNormal,
    P_High
};
```

Elements

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>P_Low</td>
<td>Low job priority.</td>
</tr>
<tr>
<td>P_BelowNormal</td>
<td>Below normal job priority.</td>
</tr>
<tr>
<td>P_Normal</td>
<td>Normal job priority.</td>
</tr>
<tr>
<td>P_AboveNormal</td>
<td>Above normal job priority.</td>
</tr>
<tr>
<td>P_High</td>
<td>High job priority.</td>
</tr>
</tbody>
</table>

See also

XmlTicket::Priority, XmlResult::Priority
RecognitionModeEnum

**RecognitionModeEnum** enumeration constants are used to set the mode of document recognition.

typedef enum {
    RM_FullPage,
    RM_BarcodesOnly
} RecognitionModeEnum;

**Elements**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RM_FullPage</td>
<td>All document content is recognized including text inside tables, pictures and barcodes (if the <a href="#">IRecognitionParams::LookForBarcodes</a> property is set to TRUE).</td>
</tr>
<tr>
<td>RM_BarcodesOnly</td>
<td>Only barcode values are extracted from the document.</td>
</tr>
</tbody>
</table>

**See also**

[IRecognitionParams::RecognitionMode](#)
RecognitionQualitySettingEnum

RecognitionQualitySettingEnum enumeration constants are used to set the recognition mode.

```c
enum RecognitionQualitySettingEnum{
    RQS_Thorough,
    RQS_Balanced,
    RQS_Fast
};
```

### Elements

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RQS_Thorough</td>
<td>The recognition will run in full mode which is optimized for quality.</td>
</tr>
<tr>
<td>RQS_Balanced</td>
<td>The recognition will run in balanced mode (an intermediate mode between full and fast modes, which specify with the help of the RQS_Thorough and RQS_Fast constants).</td>
</tr>
<tr>
<td>RQS_Fast</td>
<td>The recognition will run in fast mode which provides 2-2.5 times faster recognition speed at the cost of a moderately increased error rate (1.5-2 times more errors). On good print quality texts, ABBYY Recognition Server makes an average of 1-2 errors per page, and such moderate increase in error rate can be easily tolerated in many cases, such as full text indexing with &quot;fuzzy&quot; searches, preliminary recognition, etc.</td>
</tr>
</tbody>
</table>

**See also**

RecognitionParams::RecognitionQuality
RotationTypeEnum

**RotationTypeEnum** enumeration constants are used to set the types of rotation that to be performed upon an image.

```cpp
enum RotationTypeEnum{
    RT_NoRotation,
    RT_Automatic,
    RT_Clockwise,
    RT_Counterclockwise,
    RT_Upsidedown
};
```

**Elements**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RT_NoRotation</td>
<td>This value means no rotation.</td>
</tr>
<tr>
<td>RT_Automatic</td>
<td>Rotation will be detected automatically.</td>
</tr>
<tr>
<td>RT_Clockwise</td>
<td>Rotation 90 degrees clockwise.</td>
</tr>
<tr>
<td>RT_Counterclockwise</td>
<td>Rotation 90 degrees counterclockwise or 270 degrees clockwise.</td>
</tr>
<tr>
<td>RT_Upsidedown</td>
<td>Rotation upside down or 180 degrees.</td>
</tr>
</tbody>
</table>

**See also**

[PreprocessingParams::RotationType](https://example.com/preprocessingparams::rotationtype)
**RTFSynthesisModeEnum**

*RTFSynthesisModeEnum* enumeration constants are used to set the modes of RTF file synthesis from the recognized text when exporting to RTF format.

```cpp
enum RTFSynthesisModeEnum{
    RSM_PlainText,
    RSM_FormattedText,
    RSM_EditableCopy,
    RSM_ExactCopy
};
```

**Elements**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RSM_PlainText</td>
<td>Unlike the <em>RSM_FormattedText</em> mode, this mode does not retain formatting.</td>
</tr>
<tr>
<td></td>
<td>Retains fonts, font sizes, and paragraphs, but does not retain the exact</td>
</tr>
<tr>
<td></td>
<td>locations of the objects on the page or the spacing. The resulting text will</td>
</tr>
<tr>
<td></td>
<td>left-aligned. (Right-to-left texts will be right-aligned).</td>
</tr>
<tr>
<td></td>
<td><strong>Note.</strong> Vertical texts will be changed to horizontal in this mode.</td>
</tr>
<tr>
<td>RSM_FormattedText</td>
<td>Produces a document that nearly preserves the original format and text flow</td>
</tr>
<tr>
<td></td>
<td>but allows easy editing.</td>
</tr>
<tr>
<td>RSM_EditableCopy</td>
<td>Produces a document that maintains the formatting of the original. This</td>
</tr>
<tr>
<td></td>
<td>allows you to get the document where the position of blocks on the image</td>
</tr>
<tr>
<td></td>
<td>preserved. This option is recommended for documents with complex layout</td>
</tr>
<tr>
<td></td>
<td>such as promotion booklets. Note, however, that this option limits the</td>
</tr>
<tr>
<td></td>
<td>ability to change the text and formatting of the output document.</td>
</tr>
<tr>
<td>RSM_ExactCopy</td>
<td></td>
</tr>
</tbody>
</table>

**See also**

- *RTFExportSettings::RTFSynthesisMode*
- *MSWordExportSettings::RTFSynthesisMode*
**TextCoordinatesParticularityEnum** enumeration constants are used to specify how text should be divided: by words or by lines.

```java
enum TextCoordinatesParticularityEnum{
    TCP_Lines,
    TCP_Words
};
```

**Elements**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCP_Lines</td>
<td>The text should be divided by lines.</td>
</tr>
<tr>
<td>TCP_Words</td>
<td>The text should be divided by words.</td>
</tr>
</tbody>
</table>

**See also**

[AltoExportSettings](#)
**TextEncodingTypeEnum**

**TextEncodingTypeEnum** enumeration is the Open API internal representation of text encoding types.

```cpp
enum TextEncodingTypeEnum{
    TET_Simple,
    TET_UTF8,
    TET_UTF16
};
```

**Elements**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TET_Simple</td>
<td>Simple encoding.</td>
</tr>
<tr>
<td>TET_UTF8</td>
<td>Unicode UTF-8 format.</td>
</tr>
<tr>
<td>TET_UTF16</td>
<td>Unicode UTF-16 format.</td>
</tr>
</tbody>
</table>

**See also**

`CSVExportSettings::EncodingType`,
`TextExportSettings::EncodingType`,
`HTMLExportSettings::EncodingType`
**TextTypeEnum**

`TextTypeEnum` enumeration constants are used to describe the type of recognized text.

```java
enum TextTypeEnum{
    TT_Normal,
    TT_Typewriter,
    TT_Matrix,
    TT_OCR_A,
    TT_OCR_B,
    TT_MICR_E13B,
    TT_Gothic,
    TT_Fax
};
```

### Elements

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TT_Normal</td>
<td>This value corresponds to common typographic text.</td>
</tr>
<tr>
<td>TT_Typewriter</td>
<td>This value tells Open API to presume that the text on the image was typed on a typewriter.</td>
</tr>
<tr>
<td>TT_Matrix</td>
<td>This value tells Open API to presume that the text on the image was printed by means of a dot-matrix printer.</td>
</tr>
</tbody>
</table>
| TT_OCR_A              | This value corresponds to a monospaced font designed specifically for Optical Character Recognition. Largely used by banks, credit card companies and similar businesses. It may look as follows:  
                        | OCR A 123                                                                    |
| TT_OCR_B              | This value corresponds to a font designed specifically for Optical Character Recognition. It may look as follows:  
                        | OCR B 123                                                                    |
| TT_MICR_E13B          | This value corresponds to a special set of numeric characters printed with special magnetic inks. MICR (Magnetic Ink Character Recognition) characters are found in a variety of places, including personal checks. It may look as follows:  
                        | 0123456789...                                                              |

**Note:** If an image contains characters of a language and MICR characters, you must specify the MICR (E13B) text type and select the E13B language in addition to the main recognition language. You need not select the E13B language when recognizing English texts, because the English language already contains all E13B
characters.
This value tells Open API to presume that the text on the image was printed in Gothic type. It may look as follows:

TT_Gothic

Die Verwahrung gegen

For this text type, Open API currently supports the following fonts: Textura (Gothic script), Fraktur and Schwabacher.

TT_Fax

This value tells Open API to presume that the text on the recognized image is produced by a fax machine.

See also

RecognitionParams::TextTypes
Open API Samples

The ABBYY Recognition Server Open API samples illustrate the use of the COM-based API and Web Services API. You can find the code for all the samples in the folder `%ALLUSERSPROFILE%\Application Data\ABBYY Recognition Server 4.0\Samples`.

**Note:** In Microsoft Windows Vista, Windows 7 and Windows 8, the samples are stored in the folder `%PUBLIC%\ABBYY\ABBYY Recognition Server 4.0\Samples`.

The COM-based API samples are provided for Visual Basic, Visual Basic .Net, raw C++, C++ with the Native COM Support, and C#. The Web Services API sample is provided for C#. A description of the samples is provided in the table below.

<table>
<thead>
<tr>
<th>Sample Name</th>
<th>API</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hello</td>
<td>COM-based API</td>
<td>This sample shows the basic steps of using the ABBYY Recognition Server COM-based API.</td>
</tr>
<tr>
<td>Listening</td>
<td>COM-based API</td>
<td><em>Exists only for Visual Basic 6.0.</em> This sample shows how to use the <a href="#">Client::IsListening</a> and <a href="#">Client::StartListening</a> methods.</td>
</tr>
<tr>
<td>AsyncProcessing</td>
<td>COM-based API</td>
<td><em>Exists only for Visual Basic 6.0.</em> This sample shows the basic steps of using the ABBYY Recognition Server COM-based API for asynchronous processing.</td>
</tr>
<tr>
<td>Sample for ASP.NET</td>
<td>COM-based API</td>
<td>A sample of creating ASP.NET web-applications for online image recognition.</td>
</tr>
<tr>
<td>WebServiceHello</td>
<td>Web Services API</td>
<td><em>Exists only for C#.</em> This sample shows the basic steps of using the ABBYY Recognition Server Web Services API.</td>
</tr>
</tbody>
</table>
ABBY Recognition Server 4 COM-based API Samples
Sample Hello

This sample shows the basic steps of using the ABBYY Recognition Server COM-based API:

- creating the Client object;
- connecting to the server;
- getting the list of available workflows;
- selecting the workflow whose properties will be used for input image recognition;
- input image recognition;
- checking the recognition results.

The sample uses the following methods and properties:

<table>
<thead>
<tr>
<th>Object</th>
<th>Properties</th>
<th>Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client</td>
<td>Workflows</td>
<td>Connect,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ProcessFile</td>
</tr>
<tr>
<td>StringsCollection</td>
<td>Count</td>
<td>Item</td>
</tr>
<tr>
<td>XmlResult</td>
<td></td>
<td>IsFailed</td>
</tr>
</tbody>
</table>

The Hello sample code is intended to be used in the following development tools:

- Visual Basic 6.0
- Visual Basic.Net
- raw C++
- C++ with the Native COM Support
- C#

Note: When working with a remote host, the UNC-path to the workflow must be specified.

See also

Description of the ABBYY Recognition Server Open API Samples
ABBYY Recognition Server 4 COM-based API Samples
Sample Listening

This sample shows how to use the Client::IsListening and Client::StartListening methods of the ABBYY Recognition Server COM-based API. It performs the following:

- creating the Client object;
- connecting to the server;
- getting the list of available workflows;
- setting up notifications about the completion of work by a specified workflow.

The sample uses the following methods and properties:

<table>
<thead>
<tr>
<th>Object</th>
<th>Properties</th>
<th>Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client</td>
<td>Workflows</td>
<td>Connect,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IsListening,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OnJobComplete</td>
</tr>
<tr>
<td></td>
<td>StringsCollection</td>
<td>Count,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Item</td>
</tr>
<tr>
<td>IClientEvents</td>
<td></td>
<td>OnJobComplete</td>
</tr>
</tbody>
</table>

The Listening sample code exists only for Visual Basic 6.0.

See also

Description of the ABBYY Recognition Server Open API Samples
Sample AsyncProcessing

This sample shows the basic steps of using the ABBYY Recognition Server COM-based API for asynchronous processing. It performs the following:

- creating the Client object;
- connecting to the server;
- getting the list of available workflows;
- setting up notifications about the completion of work by a specified workflow.

The sample uses the following methods and properties:

<table>
<thead>
<tr>
<th>Object</th>
<th>Properties</th>
<th>Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client</td>
<td>Workflows</td>
<td>Connect,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ProcessFileAsync</td>
</tr>
<tr>
<td>StringCollection</td>
<td>Count</td>
<td>Item</td>
</tr>
<tr>
<td>IClientEvents</td>
<td></td>
<td>OnJobComplete</td>
</tr>
</tbody>
</table>

The AsyncProcessing sample code exists only for Visual Basic 6.0.

See also

Description of the ABBYY Recognition Server Open API Samples
ABBYY Recognition Server 4 COM-based API
Sample for ASP.NET

In the folder `%ALLUSERSPROFILE%\Application Data\ABBYY Recognition Server 4.0\Samples\ASP.NET`, you can find a sample of ASP.NET code which allows creating Web applications for online image recognition with the help of the ABBYY Recognition Server COM-based API.

**Note:** In Microsoft Windows Vista, Windows 7, and Windows 8, the sample ASP.NET code is stored in the folder `%PUBLIC%\ABBYY\ABBYY Recognition Server 4.0\Samples\ASP.NET`.

To run this sample, ASP.NET must be installed on your computer. ASP.NET is a part of Microsoft .NET Framework. To install Microsoft .NET Framework 2.0, you may run `dotnetfx20.exe` file from the DotNet folder on the Recognition Server 4 CD and follow the installation program instructions. When installing Microsoft .NET Framework, ASP.NET is copied on the computer but not installed. You should install it manually by running:

```
C:\WINDOWS\Microsoft.NET\Framework\vX.X.XXX\aspxnet_regiis.exe -i (specify the appropriate version instead of vX.X.XXX).
```

To run the ASP.NET sample application, you must do the following:

1. Install Internet Information Services (IIS) (Start > Settings > Control Panel > Add or Remove Programs > Add/Remove Windows Components).
2. Create the Hello folder in the `C:\Inetpub\wwwroot` folder.
3. Open the IIS administration console (Start > Settings > Control Panel > Administrative Tools > Internet Information Services).
4. In the Default Web Site node, create the "Hello" virtual folder (Action > New > Virtual Directory). Enter Hello in the Alias field, and enter `C:\Inetpub\wwwroot\Hello` in the Directory field. Make sure that you select the Read and Run scripts options in the Access Permissions section.
5. Open the dialog box of the properties of the created folder (Action > Properties). On the Documents tab, select the Enable Default Document option and change the default document to "Hello.aspx".
6. Copy the contents of the Samples\ASP.NET\Hello folder to the `C:\Inetpub\wwwroot\Hello` folder.
7. Make sure that the Internet Guest Account has read\write permissions for the `C:\Inetpub\wwwroot\Hello\Upload` folder and the Input folder, Output folder and Exceptions folder of the first workflow of the server.
8. Restart IIS (select Action > All Tasks > Restart IIS... in the server node menu).
9. Open the project in Microsoft Visual Studio .NET (select File > Open > Project From Web in the menu).
10. Build and start the project.

Once the application is started, a Web form appears. In the **Select images** section of this form, specify the image files. In the **Specify option** section, specify the recognition parameters. When you
click **Start Processing...**, the recognition process starts. When this process is over, you will get a page with the results.

The sample uses the following methods and properties:

<table>
<thead>
<tr>
<th>Object</th>
<th>Properties</th>
<th>Methods</th>
<th>Enumeration constants</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client</td>
<td>Workflows</td>
<td><strong>Connect</strong>,</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>CreateXmlTicket</strong>,</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>ProcessXmlTicket</strong></td>
<td></td>
</tr>
<tr>
<td>XmlTicket</td>
<td>PreprocessingParams,</td>
<td><strong>AddImage</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>RecognitionParams,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ExportParams</td>
<td></td>
<td></td>
</tr>
<tr>
<td>InputFiles</td>
<td>Count</td>
<td>Item</td>
<td></td>
</tr>
<tr>
<td>InputFile</td>
<td>OutputFiles</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RecognitionParams</td>
<td>Languages</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PreprocessingParams</td>
<td>RotationType</td>
<td></td>
<td><strong>RotationTypeEnum</strong></td>
</tr>
<tr>
<td>StringsCollection</td>
<td>Count</td>
<td>Add, Item</td>
<td></td>
</tr>
<tr>
<td>XmlResult</td>
<td>InputFiles</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

See also

[ABYY Recognition Server Open API Samples](#)
Sample WebServiceHello

This sample shows the basic steps of using the ABBYY Recognition Server Web Services API:

- creating the **RSSoapService** object;
- connecting to the server;
- getting the list of available workflows;
- selecting the workflow whose properties will be used for creating the **XmlTicket** object;
- creating the **XmlTicket** object, which represents the processing parameters of a job;
- setting processing parameters (input file, export settings);
- input image recognition;
- checking the recognition results.

**Note:** Web-service saves to the temporary files information needed to process the job. The user who starts Web-service code (usually ASPNET) must have access right to folder C:\Windows\Temp.

The sample uses the following methods and properties:

**Objects and their properties:** **RSSoapService**, **FileContainer**, **FileContainer::FileContents**, **XmlTicket**, **PDFExportSettings**, **PDFExportSettings::NamingRule**, **PDFExportSettings::PDFExportMode**, **PDFExportSettings::PictureResolution**, **PDFExportSettings::Quality**, **OutputFormatSettings**, **OutputFormatSettings::ExportParams**, **ExportParams::Formats**, **InputFile**, **InputFile::FileData**, **XmlTicket::InputFiles**, **XmlResult**, **XmlResult::IsFailed**.

**Methods:** **RSSoapService::CreateTicket**, **RSSoapService::ProcessTicket**, **RSSoapService::GetWorkflows**.

**See also**

**Description of the ABBYY Recognition Server Open API Samples**
XML Files Description Overview

This section contains:

- XML Ticket
- XML Result
- Document XML Scheme
XML Ticket

An XML Ticket is an XML file that contains the processing parameters for one or several image files. An XML Ticket may be created by the user or by the Open API. An XML Ticket must be located in the Input Folder together with the images to be processed.

XML Ticket structure is nearly identical to the structure of the `XmlTicket` object in the Open API. Most object properties have corresponding XML attributes, and the objects themselves have corresponding XML elements. Enumeration constants are specified without prefixes, for example:

```
RTFSynthesisMode = "RTFPlainText"
```

You can see an XML scheme for an XML Ticket in the `XmlTicket.xsd` file which can be found in the `Bin` subfolder of the ABBYY Recognition Server folder.

If an image is put to the Input Folder without an XML Ticket, it will be processed using the parameters specified for the workflow. If you need to process the image using parameters (e.g. recognition language, text type, etc.) different from those of the workflow, you must create an XML Ticket and put it into the Input Folder together with the image.

**Important!** You must put the XML Ticket into the Input Folder first and then put the images connected with the XML Ticket in this folder. Otherwise ABBYY Recognition Server may process the images separately from the XML Ticket.

Below follows an example of an XML Ticket which defines the following processing parameters: input images `SampleImage1.jpg` and `SampleImage2.jpg` must be converted to a single PDF file, which must be then saved in the folder `C:\Documents and Settings\All Users\Application Data\ABBYY Recognition Server 4.0\Default Workflow\Output Folder`.

```
<XmlTicket>
  <InputFile Name = "SampleImage1.jpg"/>
  <InputFile Name = "SampleImage2.jpg"/>
  <ExportParams DocumentSeparationMethod="MergeIntoSingleFile">
    <ExportFormat OutputFileFormat="PDF" OutputFlowType="SharedFolder">
      <OutputLocation>C:\Documents and Settings\All Users\Application Data\ABBYY Recognition Server 4.0\Default Workflow\Output Folder</OutputLocation>
    </ExportFormat>
  </ExportParams>
</XmlTicket>
```

See also

Com-based API: [XmlTicket Object](#), [XML Result](#)

Web Services API: [XmlTicket Object](#), [XML Result](#)
ABBYY Recognition Server 4 XML Files Description
XML Result

XML Result is an XML file created by ABBYY Recognition Server when the job has been executed. This file contains information about the job processing parameters and the job execution result. This file can be published together with output files.

XML Result structure is nearly identical to the structure of the `XmlResult` object in the Open API. Most object properties have corresponding XML attributes, and the objects themselves have corresponding XML elements. Enumeration constants are specified without prefixes, for example:

```
RTFSynthesisMode = "RTFPlainText"
```

You can see an XML scheme for an XML Result in the XmlResult.xsd file which can be found in the `Bin` subfolder of the ABBYY Recognition Server folder.

ABBYY Recognition Server first puts output images and recognition results to the Output Folder, and then creates an XML Result. This means that the appearance of an XML Result in the Output Folder can be used as an indication that the job has been completed.

See also

COM-based API: [XmlResult Object](#), [XML Ticket](#)

Web Services API: [XmlResult Object](#), [XML Ticket](#)
ABBYY Recognition Server 4 XML Files Description
When recognizing a page, ABBYY Recognition Server first analyzes its layout and detects blocks of various types on the page. Each block on the page belongs to one of the four types described below, and has its own sequence number and region (a region is a set of rectangles on the image positioned one under another in such a way that the top line of the lower rectangle is the bottom line of the upper one, so that the rectangles do not overlap). Blocks determine how and in what order the image areas should be recognized.

The following block types are supported:

**Text** - This is used for text image areas and should only contain single-column text. The recognized text is enclosed with `text` tags in the XML file. Text is represented as a set of paragraphs (each paragraph is enclosed with `par` tags). In a paragraph, each line is marked by `line` tags. For a line, formatting attributes are shown (formatting tags). Character attributes are represented in `charParams` tag attributes.

**Table** - This is used for table image areas or for areas of text that have the structure of a table. The recognized table is represented in the XML file by a set of rows (`row` tags). In a row, each cell is marked by `cell` tags. Cell text is enclosed with `text` tags.

**Picture** - This is used for image areas that contain pictures. This type of block may enclose an actual picture or any other object that should be displayed as a picture (e.g. a section of text). A picture block is only represented as a block region (`region` tags) in the XML file.

**Barcode** - This is used for barcode image areas. The recognized barcode is represented in the XML file by the barcode value (if the `LookForBarcodes` property of the RecognitionParams object is set to TRUE). The barcode value is enclosed with `text` tags.

You can see the XML scheme of an XML document in the ExportToXml.xsd file, which can be found in the Bin subfolder of the ABBYY Recognition Server folder.

**Note:** When working with a page on the Recognition Server 4 Verification Station, blocks are shown as image areas enclosed in frames of different colors, as on the picture below.

The picture below shows Picture, Text, and Table blocks in the output XML file.
Description of Tags

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Multiplicity</th>
<th>Parent Tag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>document</td>
<td>Complex Type, a sequence of page tags</td>
<td>1</td>
<td>no</td>
<td>Document</td>
</tr>
<tr>
<td>block</td>
<td>Complex Type, a sequence of BlockType tags</td>
<td>0...unbounded</td>
<td>page</td>
<td>Recognized page</td>
</tr>
<tr>
<td>region</td>
<td>Complex Type, a sequence of rect tags Has no type attributes.</td>
<td>1</td>
<td>block</td>
<td>Block region, a rectangles</td>
</tr>
<tr>
<td>rect</td>
<td>Complex Type</td>
<td>1...unbounded</td>
<td>region</td>
<td>Rectangle</td>
</tr>
</tbody>
</table>

**Type attributes**
See also

COM-based API: XMLExportSettings

Web Services API: XMLExportSettings
Important! Read the following terms carefully before installing, copying and/or otherwise using ABBYY® Recognition Server (hereinafter referred to as “the SOFTWARE”). Installing, copying or otherwise using the SOFTWARE indicates Your acceptance of these terms.

This End-User License Agreement (hereinafter referred to as the “EULA”) is a legal agreement between you, the end user, who obtained or is using the SOFTWARE, and ABBYY.

This EULA comes into force when You demonstrate Your consent to be bound by its terms by selecting the “I accept the terms of the license agreement” button followed by selecting the “Next” button and installing the SOFTWARE, or when You install, copy or start using the SOFTWARE in any manner. Any such act is deemed to demonstrate Your acknowledgment that You have read this EULA, that You understand it, and agree to be bound by its terms. If You do not agree to the terms of this EULA, do not use the SOFTWARE and disable, remove it from Your system, and destroy Your copies of the SOFTWARE. This EULA is binding for the entire period of the SOFTWARE copyright unless otherwise stated in this EULA or in a separate written agreement between You and ABBYY or an ABBYY Partner and may depend on the scope of the License as it is described in the EULA.

The SOFTWARE is protected by copyright laws and international treaty provisions and some of its portions are protected by patent and trade secret laws. You agree that this EULA is enforceable like any written negotiated agreement signed by you. This EULA is enforceable against You.

If the SOFTWARE is accompanied by a hard copy of a separate agreement with ABBYY or an ABBYY Partner, governing the terms of Your use of the SOFTWARE, then, in the case of any discrepancies in content between the text of this EULA and the text in the hard copy of such separate agreement, the text in the hard copy shall prevail.

The EULA may be available in different languages. There may be inconsistencies or differences in interpretation between the English version of the EULA and those EULAs made available in other languages. For the sake of uniformity and to avoid any ambiguity, the English version of the EULA shall govern in all disputes, claims or proceedings to interpret, enforce or otherwise relating to the EULA.

Definitions

“ABBYY” means

ABBYY USA Software House Inc., registered at 880 North McCarthy Boulevard, Suite 220, Milpitas, California, 95035, USA, when article 13.1 of this EULA applies;

ABBYY Canada Inc., registered at 555 Leggett Drive, Suite 304, Kanata, Ontario K2K 2X3 Canada, when article 13.2 of this EULA applies;
“ABBYY Partner” means an entity or individual who is licensed by ABBYY to resell and distribute licensed copies of the SOFTWARE to End Users, either directly or through one or more sub-resellers or sub-distributors.

“Computer” means a specific physical device or virtual machine that may consist of one or more CPU (central processing unit) cores and which runs a specified operating system. Any changes to the configuration or composition of the Computer could result in the Computer being considered a different Computer for licensing purposes.

“Confidential Information” means any and all information that is not generally known and that is proprietary to ABBYY and is either designated as Confidential by ABBYY or is marked either Confidential or Proprietary.

“Google Search Appliance” means the network device manufactured by Google, Inc. and intended for searching information in a local area network.

“Google Search Appliance Connector” means the SOFTWARE Component which passes the content of recognized documents to the Google Search Appliance.

“IFilter” means the SOFTWARE Component which passes the content of recognized documents to Microsoft® Office SharePoint® Server, Microsoft Search Server, Windows Search, or Desktop Search.

“Intellectual Property Rights” means all intellectual and industrial property rights and includes rights to (i) inventions, discoveries, and letters patent, including applications therefor, reissues thereof, and continuation and continuations in part; (ii) copyrights; (iii) designs and industrial designs; (iv) trademarks, service marks, trade dress and similar rights; (v) know-how, trade secrets, and confidential information; (vi) integrated circuit topography rights and rights in mask works; and (vii)
other proprietary rights.

“License” means the non-exclusive limited right granted to You by ABBYY to install and use the functionality of the SOFTWARE in accordance with the terms and conditions of this EULA.

“Open API” means the SOFTWARE Component which provides an Application Programming Interface to the SOFTWARE that allows You to call its functions from and exchange data with other Windows® applications.

“Remote Administration Console” means the SOFTWARE Component which provides a common administration interface for configuring and monitoring the SOFTWARE. The Remote Administration Console can be located on any Computer in the local area network.

“Scanning Station” means the SOFTWARE Component for managing batch scanning, viewing documents and controlling their visual quality, and managing separation and export of documents in selected file formats to selected destinations.

“Processing Station” means the Computer running the optical character recognition process.

“Verification Station” means the SOFTWARE Component for viewing and correcting the results of optical character recognition.

“Indexing Station” means the SOFTWARE Component for specifying document types and assigning document attributes.

“Script” means a set of additional operations executed within a SOFTWARE operation cycle and intended to partially automate or modify the behavior of the SOFTWARE.

“Serial Number” means a unique identifier of a License or License group with similar parameters that is being given by ABBYY or an ABBYY Partner to the End User.

“Server Computer” means the Computer running the Server Manager.

“Server Manager” means the SOFTWARE Component which manages processing options, licenses, distributes tasks among Processing Stations, Verification Stations, and Indexing Stations, and delivers output documents after the processing.

“SOFTWARE” means any version or edition of ABBYY Recognition Server, including any and all SOFTWARE Components (add-ons, custom components and modules, etc.) embedded in ABBYY Recognition Server or made available online or in any other manner, including as executable, help, demo, sample, and other files; libraries, databases, samples, associated media (images, photos, animation, audio-video components, music, etc.), printed materials, and other SOFTWARE Components.

“SOFTWARE Activation” means the procedure whereby the installed SOFTWARE is made fully operational in accordance with the SOFTWARE documentation and the scope of Your License.
“SOFTWARE Component” means any constituent part of the SOFTWARE, including, but not limited to, any computer programs contained in the SOFTWARE.

“You,” “Your,” and “End User” refer to and include any person and/or any entity that obtained this SOFTWARE for its own use and not for purposes of further resale and any actual user of the SOFTWARE.

1. License Grant

1.1 Subject to the terms and conditions of the EULA, ABBYY grants to You a limited, non-exclusive License to install and use the functionality of the SOFTWARE, subject to all restrictions and the scope of the License as may be provided by this EULA, the software and/or hardware license keys provided by ABBYY (“License Key”), the SOFTWARE, and/or a separate written agreement between You and ABBYY or an ABBYY Partner. All provisions stated herein apply both to the SOFTWARE as a whole and to all of its separate components and end user documentation with the exception of the third party technologies specified in Article 12, which are covered by their own licenses. Any issue with respect to the scope of the license shall be construed in favour of restrictions on the scope of the license. The License Key technically restricts the use of the SOFTWARE and reflects the scope of the License You obtained. Restrictions on Your use of the SOFTWARE and the scope of Your License may include but are not limited to the following:

1.1.1 Number of SOFTWARE Components, seats and network access. Information about the number of SOFTWARE Components and Computers on a Network allowed for simultaneous work with the SOFTWARE is specified in Your License Key and/or written agreement between You and ABBYY or an ABBYY Partner and may include but is not limited to the following:

1.1.1.1 You may install, run, and use one (1) Server Manager on one (1) Server Computer.

1.1.1.2 You may install the Scanning Station on an unlimited number of Computers. The number of Scanning Stations used simultaneously may not exceed the number specified in Your License Key.

1.1.1.3 You may install the Verification Station on an unlimited number of Computers. The number of Verification Stations used simultaneously may not exceed the number specified in Your License Key.

1.1.1.4 You may install the Indexing Station on an unlimited number of Computers. The number of Indexing Stations used simultaneously may not exceed the number specified in Your License Key.

1.1.1.5 You may install and run one (1) Google Search Appliance Connector on one (1) Server Computer in order to use the SOFTWARE in conjunction with the Google Search Appliance if this is allowed by Your License Key.

1.1.1.6 You may install and run IFilter on an unlimited number of Computers in order to use the SOFTWARE in conjunction with Windows Search or Desktop Search, Microsoft Office SharePoint Server or Microsoft Search Server if this is allowed by Your License Key.

1.1.1.7 You may install, run, and use the Open API and the Remote Administration Console on an
unlimited number of Computers.

1.1.2 Processing volume. The number of volume units (e.g. pages) that may be processed by the SOFTWARE may be limited in one or more ways, such as limitations on the number of volume units that may be processed within a particular time period, such as monthly or annually, or on the overall number of volume units that may be processed. The size of a page or other volume units may also be limited. Processing volume limitations are specified in Your License Key and/or written agreement between You and ABBYY or an ABBYY Partner.

1.1.3 Duration. Use of the SOFTWARE may be limited by a fixed period of time, and the SOFTWARE may not be used beyond the expiration of such time period. Limitation of the duration is specified in Your License Key and/or a written agreement between You and ABBYY or an ABBYY Partner.

1.1.4 Features, add-ons and custom components. The use of particular features, add-ons and/or custom components of the SOFTWARE may be limited by Your License Key and/or a written agreement between You and ABBYY or an ABBYY Partner.

1.1.5 Processing power. The total number of CPU cores used by the SOFTWARE on the Processing Stations may not exceed the number specified in Your License Key and/or a written agreement between You and ABBYY or an ABBYY Partner.

1.2 You may use the SOFTWARE as a part of or in conjunction with any external application only if You have the right to use that application in such a manner.

1.3 All rights not expressly granted to You by this EULA are reserved by ABBYY. This EULA does not grant You any rights in connection with any trademarks of ABBYY.

1.4 If Article 13.6 applies and You are an individual human being person, You may use the SOFTWARE worldwide. If Article 13,6 applies and You are an entity, the SOFTWARE may be obtained (purchased) only in the country where the entity or its branch and representative offices are registered, unless otherwise agreed in a separate written agreement between You and ABBYY or an ABBYY Partner. The employees of the entity or of its branch and representative offices may use the SOFTWARE worldwide, provided the SOFTWARE has been obtained and installed in the country where the entity or its branch and representative offices are registered, unless otherwise agreed in a separate written agreement between You and ABBYY or an ABBYY Partner.

1.5 Any use of the SOFTWARE or component parts outside of or in contravention of the terms and conditions of this EULA shall constitute a breach of ABBYY’s and/or third parties’ intellectual property rights and shall give cause for the revocation of all rights to use the SOFTWARE granted to You under this EULA.

1.6 If You deploy or use the SOFTWARE in a virtual environment, at no time may access or use of the SOFTWARE exceed restrictions on the SOFTWARE, or the scope of the license that has been granted. For instance, the same License Key may not be used to permit use or access to the SOFTWARE in a virtual environment by a number of Computers that is greater than the number of
Computers for which a license has been properly obtained; the SOFTWARE may not be used to process more than the permitted number of volume units; and the number of SOFTWARE copies running in a virtual environment may not exceed the number of copies for which a license has been properly obtained.

1.7 If You access the SOFTWARE by using terminal services that allow accessing the SOFTWARE on a remote computer over a network, the amount of simultaneously running instances of the SOFTWARE may not exceed the number of Computers for which a License has been obtained.

1.8 You may receive the SOFTWARE on more than one medium (multiple-media SOFTWARE), including downloads over the Internet. Regardless of the quantity or type of media You receive, You are only licensed to use one (1) Server Manager on one (1) Server Computer subject to this EULA.

2. Limitations of Use

2.1 You may not be able to exercise Your rights to use the SOFTWARE under this EULA unless You perform SOFTWARE Activation.

2.2 All terms of use and limitations governing the use of the SOFTWARE are stated in this EULA, unless otherwise stipulated in a separate written agreement between You and ABBYY or an ABBYY Partner or in other documentation accompanying the SOFTWARE.

2.3 You may not perform or make it possible for other persons to perform any activities included in the list below:

2.3.1 Reverse engineer, disassemble, decompile (i.e. reproduce and transform the object code into source code) or otherwise attempt to derive the source code for the SOFTWARE or any part, except, and only to the extent, that such activity is expressly permitted by applicable law notwithstanding this limitation. If applicable law prohibits the restriction of such activities, any information so discovered shall not be disclosed to third parties unless such disclosure is required by law and such information shall be promptly disclosed to ABBYY. All such information shall be deemed to be confidential and proprietary information of ABBYY.

2.3.2 Modify, adapt (including any changes for the purpose of enabling the SOFTWARE to run on Your hardware), or make any changes to the object code of the Software, applications and databases contained in the SOFTWARE other than those provided for by the SOFTWARE and described in the documentation.

2.3.3 Correct errors in the SOFTWARE or translate the SOFTWARE without the prior written consent of ABBYY.

2.3.4 Rent, lease, sublicense, assign or transfer any rights granted to You by this EULA and other rights related to the SOFTWARE to any other person or authorize all or any portion of the SOFTWARE to be copied onto another computer (except as described in article 2.6) unless otherwise stipulated in a separate written agreement with ABBYY or an ABBYY Partner.
2.3.5 Make it possible for any person not entitled to use the SOFTWARE to access and/or use the SOFTWARE, including without limitation in a multi-user system, virtual environment, or via the Internet.

2.3.6 Remove, change or obscure any copyright, trademark or patent notices that appear on the SOFTWARE as delivered to You.

2.4 You may not use the SOFTWARE to provide to any third party paid or free recognition or conversion services and/or to provide the results or access to the results acquired through the use of the SOFTWARE as a part of another service that has recognition or conversion as its component unless You have signed a separate written agreement with ABBYY.

2.5 You may not bypass the SOFTWARE’s user interface that is provided with the SOFTWARE unless otherwise stipulated in an additional written agreement with ABBYY.

2.6 You may only make a one-time permanent transfer of this SOFTWARE directly to another end user. If You are a legal entity, such a transfer requires written approval by ABBYY. This transfer must include all of the SOFTWARE (including all SOFTWARE Components, the media and printed materials, any upgrades and/or updates, and this EULA). Such transfer may not be by way of consignment or any other indirect transfer. The transferee of such one-time transfer must agree to comply with the terms of this EULA, including the obligation not to further transfer this SOFTWARE. You must uninstall the SOFTWARE from all of Your Computers or from Your Local Area Network in the case of such a SOFTWARE transfer.

3. Pre-Release and Trial SOFTWARE

3.1 If the SOFTWARE for which You have received a License is: (a) pre-commercial release or Beta-Release Software (“Pre-Release SOFTWARE”); or (b) SOFTWARE that is labeled “Try&Buy,” “Trial,” “Demo” or was provided at no charge for trial demonstration, evaluation or testing purposes (collectively “Trial SOFTWARE”), then this article 3 (Pre-Release and Trial SOFTWARE) shall apply until such time as You obtain (purchase) a commercial use License for the SOFTWARE. “Restricted Software” shall mean jointly the Pre-Release SOFTWARE and the Trial SOFTWARE.

3.2 To the extent that any provision of this article 3 (Pre-Release and Trial SOFTWARE) conflicts with any other term or condition in this EULA, this article shall govern and shall supersede such other term or condition with respect to the Restricted SOFTWARE.

3.3 NO WARRANTY. NO ORAL, RECORDED, AND/OR WRITTEN INFORMATION OR ADVICE GIVEN BY ABBYY OR AN ABBYY AUTHORIZED REPRESENTATIVE SHALL CREATE A WARRANTY. THE RESTRICTED SOFTWARE (INCLUDING ITS TECHNICAL DOCUMENTATION) IS DELIVERED "AS IS" AND WITHOUT WARRANTY OF ANY KIND. ABBYY DOES NOT WARRANT THAT THE RESTRICTED SOFTWARE IS ERROR FREE OR THAT IT WILL WORK IN ALL CIRCUMSTANCES. ABBYY DISCLAIMS ALL WARRANTIES AND INDEMNITIES WITH RESPECT TO THE RESTRICTED SOFTWARE, WHETHER EXPRESS, IMPLIED OR STATUTORY, INCLUDING BUT NOT LIMITED TO
IMPLIED WARRANTIES OF MERCHANTABILITY, SATISFACTORY QUALITY, FITNESS FOR A PARTICULAR PURPOSE, ACCURACY, TITLE AND/OR NON-INFRINGEMENT OF THIRD PARTY RIGHTS.

3.4 LIMITATION OF LIABILITY. USE OF THE RESTRICTED SOFTWARE IS AT YOUR SOLE RISK AND YOU ASSUME THE ENTIRE RISK OF ANY AND ALL CONSEQUENCES RELATING TO THE RESTRICTED SOFTWARE. TO THE GREATEST EXTENT PERMITTED BY APPLICABLE LAW, IN NO EVENT SHALL ABBYY BE LIABLE FOR PERSONAL INJURY, OR ANY DIRECT, INCIDENTAL, SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES WHATSOEVER, INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF PROFITS, LOSS OF DATA, BUSINESS INTERRUPTION OR ANY OTHER DAMAGES OR LOSSES, ARISING OUT OF OR RELATED TO YOUR USE OR INABILITY TO USE THE RESTRICTED SOFTWARE, HOWEVER CAUSED, REGARDLESS OF THE THEORY OF LIABILITY (CONTRACT, TORT OR OTHERWISE) AND EVEN IF ABBYY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. WHERE LIABILITY CANNOT BE EXCLUDED BUT MAY BE LIMITED, ABBYY’S TOTAL LIABILITY, AND THE TOTAL LIABILITY OF AN ABBYY PARTNER, SHALL BE LIMITED TO THE SUM OF FIFTY UNITED STATES DOLLARS (U.S. $50) OR THE TOTAL AMOUNT YOU PAID FOR THE RESTRICTED SOFTWARE, WHICHEVER IS GREATER.

3.5 Restricted Software may have limited functionality, such as having functionality for a limited period of time, and upon expiration of permitted functionality (“time-out”), Your access and ability to use the Restricted Software may be disabled. Upon the occurrence of a time-out, Your rights under the EULA shall terminate, unless You obtain a new license from ABBYY.

3.6 If You have been provided the Restricted Software pursuant to a separate written agreement Your use of the SOFTWARE is also governed by such agreement. To the extent that any term or condition of a separate written agreement are in conflict with any term or condition of this EULA, a separate written agreement shall supersede such other term(s) and condition(s) with respect to the SOFTWARE, but only to the extent necessary to resolve the conflict.

3.7 Trial SOFTWARE.

3.7.1 You may use Trial SOFTWARE only for Your internal evaluation and testing purposes. Notwithstanding anything to the contrary in Section 2.6 You may not transfer or re-distribute the Trial SOFTWARE.

3.7.2 You shall not use any results or outputs acquired using the Trial SOFTWARE in regular business activities and/or in a commercial operating environment, including without limitation in a way that can generate direct or indirect revenue.

3.7.3 The Trial SOFTWARE has limited functionality and Your access to the functionality of the Trial SOFTWARE is restricted in duration by a License key. The expiration of the License key will render the Trial Software unusable and may render any material prepared using the Trial SOFTWARE inaccessible until You have purchased a commercial use license to the SOFTWARE.
3.7.4 No maintenance or technical support is provided for Trial SOFTWARE.

3.8 Pre-Release SOFTWARE.

3.8.1 Pre-Release SOFTWARE constitutes the Confidential Information of ABBYY.

3.8.2 As to Pre-Release SOFTWARE ABBYY does not promise or guarantee, has no express or implied obligation, and You acknowledge that ABBYY has not promised or guaranteed, that such software will be further developed, or that a commercial version will be announced or made available in the future. ABBYY may not introduce a software product similar to or compatible with such software. Accordingly, You acknowledge that any use, research or development that You perform regarding such software is done entirely at Your own risk.

3.8.3 You agree to provide feedback to ABBYY regarding testing and use of the Pre-Release SOFTWARE, including error or bug reports (“Feedback”), and You transfer and grant to ABBYY free of charge all of Your rights, title, and interest in such Feedback, including without limitation, the rights to use, publish and disseminate the Feedback. All Feedback constitutes the Confidential Information of ABBYY.

3.8.4 Confidentiality of Pre-Release SOFTWARE and its results.

3.8.4.1. You agree not to disclose accompanying written, oral or electronic information divulged to You by ABBYY related to the Pre-Release SOFTWARE, any information about the quality of such Pre-Release SOFTWARE or the quality of the results acquired through the use of such Pre-Release SOFTWARE, Feedback, and any information about bugs, errors and other problems discovered by You in the Pre-Release SOFTWARE are the Confidential Information of ABBYY.

3.8.4.2. You shall not disclose Confidential Information. The term “disclose” means to display, describe, copy, lease, loan, rent, assign, transfer or provide access, over a network or otherwise, to Confidential Information reproduced in any form, including oral communications, to any third party.

3.8.4.3. You shall take all reasonable steps to prevent the disclosure of Confidential Information and to keep it confidential.

3.8.5 You shall promptly inform ABBYY if You become aware of any disclosure of Confidential Information. If You are in breach of the terms and conditions set forth in articles 3.8.4.1 - 3.8.4.3 above, You shall compensate ABBYY for any loss resulting from such breach.

3.8.6. Upon receipt of a later version of Pre-Release SOFTWARE or a commercial release of SOFTWARE, whether as a stand-alone product or as part of a larger product, You agree to return or destroy all earlier versions of the Pre-Release SOFTWARE received from ABBYY.

4. Not-for-resale SOFTWARE

4.1 If the SOFTWARE is labeled “Not for Resale” or “NFR,” then, notwithstanding other articles of this EULA, You may only use such SOFTWARE for demonstration, verification or testing purposes.
5. Updates

5.1 If the SOFTWARE is labeled as an update, You must own a license for the previous version of the SOFTWARE identified by ABBYY as being eligible for this update in order to use the SOFTWARE.

5.2 The SOFTWARE labeled as an update replaces and/or supplements the product that formed the basis for Your eligibility for the update.

5.3 You may only use the resulting updated product in accordance with the terms of the EULA supplied with this update.

5.4 You acknowledge that any obligation ABBYY may have to support the version of the SOFTWARE being updated shall end upon the availability of the update.

6. Technical Support and Maintenance

6.1 You may be provided with technical support and maintenance related to the SOFTWARE (“Support Services”) subject to the current ABBYY support policy. However, You may be entitled to a different level of Support Services than is stated in the ABBYY support policy in accordance with a written agreement executed between You and ABBYY or an ABBYY Partner with respect to such Support Services.

6.2 General terms and conditions of the ABBYY support policy are published on the ABBYY Web site www.abbyy.com. ABBYY reserves the right to change the support policy at any time without any prior notice.

6.3 In addition to the general terms and conditions, ABBYY may have specific support policies in specific regions which may be regulated by separate agreements.

6.4 Any supplementary software code and any SOFTWARE component provided to You as part of Support Services is to be considered a part of the SOFTWARE and subject to the terms and conditions of this EULA.

6.5 To be eligible for Support Services, You may be required to provide ABBYY with information about the characteristics of Your hardware, the Serial Number of Your SOFTWARE as well as certain personal information, such as Your name, the name of Your company (if applicable), address, phone number and e-mail address. ABBYY may use the above-mentioned information as described in article 15.2 of the EULA.

7. Ownership.

7.1 No title to any of ABBYY’s Intellectual Property Rights (including the SOFTWARE and any ABBYY patents, trademarks, or copyrights) is transferred to You. You shall not, in any way, during or after the term of the Agreement, make any use of or claim any right to any name, logo, trademark, pattern, or design owned by ABBYY, or any name, logo, trademark, pattern, or design resembling them.
7.2 No title to any of Your Intellectual Property Rights (including the application and any of Your patents, trademarks, or copyrights) is transferred to ABBYY.

7.3 The SOFTWARE contains valuable trade secrets and confidential information belonging to ABBYY and third parties and is protected by copyright laws, including, without limitation, by United States Copyright Law, laws of Russian Federation, international treaty provisions, and the applicable laws of the country in which it is being used or obtained.

7.4 All title and rights in and to the content that is not contained in the SOFTWARE, but may be accessible through the use of the SOFTWARE, are the property of the respective content owners and may be protected by applicable copyright or other intellectual property laws and international treaties. This EULA does not grant You any rights to such intellectual property.


8.1 If required by the legislation of the country in which You obtained (purchased) the SOFTWARE, ABBYY warrants that the media on which the SOFTWARE is furnished will be free from defects in materials and workmanship under normal use only and only for the minimal guarantee term determined by the legislation of the country in which You obtained (purchased) the SOFTWARE starting from the date of purchase. If the SOFTWARE was obtained (purchased) in the countries defined in article 13.6, this period will constitute thirty (30) days starting from the date You obtained (purchase) the SOFTWARE.

8.2 EXCEPT AS EXPRESSLY PROVIDED IN THIS ARTICLE 8 (LIMITED WARRANTY, DISCLAIMERS), AND EXCEPT FOR ANY WARRANTY, CONDITION, REPRESENTATION, OR TERM TO THE EXTENT TO WHICH THE SAME CANNOT OR MAY NOT BE EXCLUDED OR LIMITED BY LAW APPLICABLE TO YOU IN YOUR JURISDICTION, THE SOFTWARE, INCLUDING WITHOUT LIMITATION ANY SOFTWARE DOCUMENTATION, UPGRADES AND UPDATES IS BEING DELIVERED TO YOU “AS IS” AND ABBYY MAKES NO WARRANTIES, CONDITIONS, REPRESENTATIONS OR TERMS (EXPRESS OR IMPLIED WHETHER BY STATUTE, COMMON LAW, CUSTOM, USAGE OR OTHERWISE) AS TO ANY MATTER, INCLUDING WITHOUT LIMITATION NON-INFRINGEMENT OF THIRD PARTY RIGHTS, MERCHANTABILITY, INTEGRATION, SATISFACTORY QUALITY, OR FITNESS FOR ANY PARTICULAR PURPOSE, OR THAT THE SOFTWARE WILL CONTAIN NO ERRORS, MEET YOUR REQUIREMENTS, OR THAT THE SOFTWARE WILL FUNCTION PROPERLY WHEN USED IN CONJUNCTION WITH ANY OTHER SOFTWARE OR HARDWARE. ABBYY DOES NOT AND CANNOT WARRANT THE PERFORMANCE OR RESULT YOU MAY OBTAIN BY USING THE SOFTWARE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE SOFTWARE LIES WITH YOU. FURTHERMORE ABBYY MAKES NO WARRANTIES FOR ANY THIRD PARTY SOFTWARE PRODUCTS WHICH MAY BE SUPPLIED WITHIN THE SOFTWARE.

9. Limitation of Liability
9.1 In no event will ABBYY be liable to you for any damages, business interruption, loss of data or information of any kind, business or otherwise, claims or costs whatsoever, or any consequential, indirect, incidental, special or punitive damages, or any lost profits or lost revenue resulting from and/or relating to the use of the software, or damages caused by possible errors or misprints in the software, even if an ABBYY representative has been advised of the possibility of such loss, damages, claims or costs, or for any claim by any third party. The foregoing limitations and exclusions apply to the extent permitted by applicable law in your jurisdiction. ABBYY’s sole and aggregate liability under or in connection with the software shall be limited to the purchase price originally paid for the software, if any.

10. Warranty Exclusion and Limitations for Users Residing in Germany or Austria

10.1 If you obtained your copy of the SOFTWARE in Germany or Austria, and you usually reside in such country, then, in accordance with German law, ABBYY warrants that the SOFTWARE provides the functionalities set forth in its documentation included in the SOFTWARE (the “agreed upon functionalities”) for the limited warranty period following receipt of the SOFTWARE copy when used on the recommended hardware configuration. As used in this article, “limited warranty period” means one (1) year if you are a business user or legal entity, and two (2) years if you are not a business user. Non-substantial variation from the agreed upon functionalities shall not be considered and does not establish any warranty rights. This limited warranty does not apply to the SOFTWARE provided to you free of charge, for example, updates, pre-release versions, “Trial” versions, product samples, “Not for resale” (“NFR”) copies of the SOFTWARE, or SOFTWARE that has been altered by you, to the extent such alterations caused a defect. To make a warranty claim, you must return, at our expense, the SOFTWARE and proof of purchase during the limited warranty period to the location where you obtained it. If the functionalities of the SOFTWARE vary substantially from the agreed upon functionalities, ABBYY is entitled (by way of re-performance and at its own discretion) to repair or replace the SOFTWARE. If this fails, you are entitled to a reduction of the purchase price (reduction) or to cancel the purchase agreement (rescission). For further warranty information, please contact ABBYY’s Customer Support Department in Germany: ABBYY Europe GmbH, Elsheimerstrasse 49, 80687 Munich, tel.: +49 (0)89 51 11 590, fax: +49 (0)89 51 11 5959.

10.2 If you obtained your copy of the SOFTWARE in Germany or Austria, and you usually reside in one of these countries, then:

10.2.1 Subject to the provisions in article 10.2.2, ABBYY’s statutory liability for damages shall be limited as follows: (i) ABBYY shall be liable only up to the amount of damages as typically foreseeable at the time of entering into the purchase agreement in respect of damages caused by a slightly negligent breach of a material contractual obligation and (ii) ABBYY shall not be liable for damages caused by a slightly negligent breach of a non-material contractual obligation.

10.2.2 The limitation of liability set forth in article 10.2.1 shall not apply to any mandatory statutory
liability, in particular, to liability under the German Product Liability Act, liability for assuming a specific guarantee or liability for culpably caused personal injuries.

10.2.3 You are required to take all reasonable measures to avoid and reduce damages, in particular to make backup copies of the SOFTWARE and Your computer data subject to the provisions of this EULA.

11. Additional Limitations for SOFTWARE obtained in the USA

11.1 **Government Use.** If use is made of the SOFTWARE by the United States Government or any US Government agency, the following additional terms shall apply: (1) Restricted Computer Software, as defined in the Rights in Data-General clause at Federal Acquisition Regulations 52.227-14; and (2) any use, duplication, or disclosure by the Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013.

11.2 **Export rules.** You agree that You shall not export or re-export the SOFTWARE in violation of any export provisions in the laws of the country in which this SOFTWARE was purchased or otherwise acquired. In addition, You represent and warrant that You are not prohibited under applicable laws from receiving the SOFTWARE.

12. Third-party technologies

12.1 **Embedded Fonts**

12.1.1 Font programs are subject to copyright, and the copyright owner may impose conditions under which a font program can be used. One of the conditions may be that you need a licensed copy of the font program to embed the font into a PDF file. In no event shall ABBYY be liable for any damages arising out of or in connection with your use of embedded fonts.

12.2 Software Datalogics and Adobe Products

12.2.1 Copyright 2000 - 2012 Datalogics, Inc.

Copyright 1984 - 2012 Adobe Systems Incorporated and its licensors. All rights reserved.

12.2.2 The term "Software Datalogics and Adobe Products" means Software Datalogics and third party products (including Adobe Products) and related documentation, and any upgrades, modified versions, updates, additions, and copies thereof.

12.2.3 You acknowledge and agree that third party licensors are the owners of certain proprietary information and intellectual property rights included in the Host Products and the Documentation. Such third party licensors are third party beneficiaries entitled to enforce ABBYY’s rights and your obligations hereunder and to seek appropriate legal and equitable remedies, including but not limited to, damages and injunctive relief, for your breach of such obligations.
12.2.4 License Grant and Restrictions. ABBYY grants you a non-exclusive right to use the Software Datalogics and Adobe Products under the terms of this EULA. You may make one backup copy of the Software Datalogics and Adobe Products, provided the backup copy is not installed or used on any computer.

12.2.5 Intellectual Property Rights. The Software Datalogics and Adobe Products incorporated into the SOFTWARE are owned by Datalogics, Adobe and third parties and their suppliers, and their structure, organization and code are the valuable trade secrets of Datalogics, Adobe and third parties and their suppliers. Software Datalogics and Adobe Products are also protected by United States Copyright law and International Treaty provisions. You may not copy the Software Datalogics and Adobe Products, except as provided in this EULA. Any copies that you are permitted to make pursuant to this EULA must contain the same copyright and other proprietary notices that appear on or in the Software Datalogics and Adobe Products. You agree not to modify, adapt, translate, reverse engineer, decompile, disassemble or otherwise attempt to discover the source code of the Software Datalogics and Adobe Products. Except as stated above, this EULA does not grant you any intellectual property rights in the Software Datalogics and Adobe Products.

12.2.6 Font License. You may embed copies of the font software into your electronic documents for purpose of printing, viewing and editing the document. No other embedding rights are implied or permitted under this license.

12.2.7 Warranty. ABBYY AND ITS SUPPLIERS DO NOT AND CANNOT WARRANT THE PERFORMANCE RESULTS YOU MAY OBTAIN BY USING THE SOFTWARE.

12.2.8 Export Rules. You agree that the Software Datalogics and Adobe Products will not be shipped, transferred or exported into any country or used in any manner prohibited by the United States Export Administration Act or any other export laws, restrictions or regulations (collectively the “Export Laws”). In addition, if the Software Datalogics and Adobe Products are identified as export-controlled items under the Export Laws, you represent and warrant that you are not a citizen, or otherwise located within, an embargoed nation and that you are not otherwise prohibited under the Export Laws from receiving the Software Datalogics and Adobe Products. All rights to use the Software Datalogics and Adobe Products are granted on condition that such rights are forfeited if you fail to comply with the terms of this EULA.

12.2.9 Governing Law. This EULA will not be governed by the conflict of law rules of any jurisdiction or the United Nations Convention on Contracts for the International Sale of Goods, the application of which is expressly excluded.

12.2.10 General Provisions. This EULA shall not prejudice the statutory rights of any party dealing as a consumer.

12.2.11 Trademarks. Trademarks appearing herein are either registered trademarks or trademarks of their respective owners in the United States and/or other countries.

12.3 PixTools® Scan Runtime Libraries.
12.3.1 "EMC Software" shall mean PixTools Scan runtime libraries, Level 1 ISIS Drivers, Level 2 ISIS Drivers, and Level 3 ISIS Drivers. The SOFTWARE may use the EMC Software for communicating between scanner and SOFTWARE through the ISIS interface.

12.3.2 License Grant and Restrictions. ABBYY grants you a non-exclusive right to use the EMC Software incorporated into the SOFTWARE under the terms of this EULA.

12.3.3 Intellectual Property Rights. The EMC Software incorporated into the SOFTWARE is owned by EMC Corporation and its suppliers, and its structure, organization and code are the valuable trade secrets of EMC Corporation and its suppliers. The EMC Software is protected by United States Copyright Law and International Treaty provisions. You may not copy the EMC Software incorporated into the SOFTWARE. You agree not to modify, adapt, translate, reverse engineer, decompile, disassemble or otherwise attempt to discover the source code of the EMC Software incorporated into the SOFTWARE. Except as stated above, this EULA does not grant you any intellectual property rights in the EMC Software.

12.3.4 Warranty. ABBYY DOES NOT AND CANNOT WARRANT THE PERFORMANCE RESULTS YOU MAY OBTAIN BY USING THE EMC SOFTWARE INCORPORATED INTO THE SOFTWARE.

IN NO EVENT WILL EMC CORPORATION OR ITS SUPPLIERS BE LIABLE TO YOU FOR ANY CONSEQUENTIAL, INCIDENTAL OR SPECIAL DAMAGES, INCLUDING ANY LOST PROFITS OR LOST SAVINGS OR FOR ANY CLAIM BY ANY THIRD PARTY.

Some states or jurisdictions do not allow the exclusion or limitation of incidental, consequential or special damages, so the above limitations may not apply to you. You may have other rights, which vary from state to state or jurisdiction to jurisdiction.

12.3.5 Trademarks. ISIS and PixTools are either registered trademarks or trademarks of EMC Corporation in the United States and/or other countries.

12.4 Cuminas DjVu® SDK.

12.4.1 You, as the End User, have the right to use DjVu SDK as part of the SOFTWARE for the entire period of the copyright for such SOFTWARE or for another lesser term of use stated in the EULA.

12.4.2 License terms. You may not perform or make it possible for other persons to perform reverse engineering, disassembly, decompilation (i.e. reproduction and transformation of the object code into source code) or otherwise attempts to derive the source code for the SOFTWARE or any part thereof, except, and only to the extent, that such activities are expressly permitted by applicable law notwithstanding this limitation. If applicable law prohibits the restriction of such activities, any information so discovered must not be disclosed to third parties with the exception that such disclosure is required by law and such information must be promptly disclosed to ABBYY. All such information shall be deemed to be confidential and proprietary information of ABBYY.

12.4.3 ABBYY MAKES NO WARRANTIES, CONDITIONS, REPRESENTATIONS OR TERMS (EXPRESS OR IMPLIED WHETHER BY STATUTE, COMMON LAW, CUSTOM, USAGE OR OTHERWISE) AS TO ANY MATTER, INCLUDING WITHOUT LIMITATION NON-
INFRINGEMENT OF THIRD PARTY RIGHTS, MERCHANTABILITY, INTEGRATION, SATISFACTORY QUALITY, OR FITNESS FOR ANY PARTICULAR PURPOSE, OR THAT THE SOFTWARE WILL CARRY NO ERRORS, MEET YOUR REQUIREMENTS, OR THAT THE SOFTWARE WILL FUNCTION PROPERLY WHEN USED IN CONJUNCTION WITH ANY OTHER SOFTWARE OR HARDWARE. ABBYY DOES NOT AND CANNOT WARRANT THE PERFORMANCE OR RESULT YOU MAY OBTAIN BY USING THE SOFTWARE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE SOFTWARE LIES WITH YOU. FURTHERMORE ABBYY MAKES NO WARRANTIES FOR ANY THIRD PARTY SOFTWARE PRODUCTS WHICH MAY BE SUPPLIED WITHIN THE SOFTWARE.

12.4.4 IN NO EVENT WILL ABBYY BE LIABLE TO YOU FOR ANY DAMAGES, BUSINESS INTERRUPTION, LOSS OF DATA OR INFORMATION OF ANY KIND, BUSINESS OR OTHERWISE, CLAIMS OR COSTS WHATSOEVER, OR ANY CONSEQUENTIAL, INDIRECT, INCIDENTAL DAMAGES, OR ANY LOST PROFITS OR LOST SAVINGS RESULTING FROM AND/OR RELATING TO THE USE OF THE SOFTWARE, OR DAMAGES CAUSED BY POSSIBLE ERRORS OR MISPRINTS IN THE SOFTWARE, EVEN IF AN ABBYY REPRESENTATIVE HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH LOSS, DAMAGES, CLAIMS OR COSTS, OR FOR ANY CLAIM BY ANY THIRD PARTY. THE FOREGOING LIMITATIONS AND EXCLUSIONS APPLY TO THE EXTENT PERMITTED BY APPLICABLE LAW IN YOUR JURISDICTION. ABBYY’S SOLE AND AGGREGATE LIABILITY UNDER OR IN CONNECTION WITH THIS EULA SHALL BE LIMITED TO THE PURCHASE PRICE ORIGINALLY PAID FOR THE SOFTWARE, IF ANY.

12.4.5 Limitations for SOFTWARE obtained in the USA.

a. Government Use. If use is made of the SOFTWARE by the United States Government or any US Government agency, the following additional terms shall apply: (1) Restricted Computer Software, as defined in the Rights in Data-General clause at Federal Acquisition Regulations 52.227-14; and (2) any use, duplication, or disclosure by the Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013.

b. Export rules. You agree that You shall not export or re-export the SOFTWARE in violation of any export provisions in the laws of the country in which this SOFTWARE was purchased or otherwise acquired. In addition, You represent and warrant that You are not prohibited under applicable laws from receiving the SOFTWARE.

12.4.6 Third-Party Code:

12.4.6.1 Oniguruma Copyright © 2002-2006 K.Kosako . All rights reserved. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met: Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution. THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS “AS IS” AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

12.4.6.2 MD5. A portion of this software is derived from the RSA Data Security, Inc. MD5 Message-Digest Algorithm. Mersenne Twister Copyright © 2006, 2007 Mutsuo Saito, Makoto Matsumoto and Hiroshima University. All rights reserved. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met: Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution. Neither the name of the Hiroshima University nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission. THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS “AS IS” AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

12.5 Wibu-Systems

12.5.1 WIBU, CodeMeter, SmartShelter, and SmartBind are registered trademarks of WIBU-SYSTEMS AG.

12.6 Other Licensed Technologies

12.6.1 This SOFTWARE is used under license to U.S. Patent No. 5,768,416.

13. Governing Law

13.1 If the SOFTWARE was obtained in the United States, Mexico, Belize, Costa Rica, El Salvador, Guatemala, Honduras, Montserrat, Nicaragua, Panama, Turks and Caicos Islands, Virgin Islands or Taiwan, this EULA shall be governed by and construed in accordance with the laws of the State of
California, United States of America. With respect to any dispute which may arise in connection with this EULA and/or this SOFTWARE, You consent to the exclusive jurisdiction and venue of the federal and/or state courts in the county of Santa Clara in the state of California. For the avoidance of doubt if the SOFTWARE was obtained in the United States, You obtained (purchased) the SOFTWARE from ABBYY USA Software House, Inc.

13.2 If the Software was obtained in Canada, this EULA shall be governed by and construed in accordance with the laws of the Province of Ontario and the Country of Canada and the parties accept the exclusive jurisdiction and venue of the federal and/or provincial courts in the Province of Ontario.

13.3 If the Software was obtained in Japan, this EULA shall be governed by and construed in accordance with the laws of Japan and the parties accept the exclusive jurisdiction of the Tokyo District Court of Japan.

13.4 If the SOFTWARE was obtained in Austria, Belgium, Denmark, Finland, France, Germany, Italy, Luxembourg, the Netherlands, Portugal, Spain, Sweden, or any other member state of the European Union, except for Malta, Greece and Cyprus, that is not directly mentioned in article 13.5, 13.6 or 13.7 of this EULA, or in Switzerland, Norway, Iceland or Liechtenstein, this EULA shall be governed by and construed in accordance with the substantive law in force in Munich, the Federal Republic of Germany and the competent court of Munich, the Federal Republic of Germany shall have exclusive jurisdiction over all disputes relating to this EULA.

13.5 If the SOFTWARE was obtained in the United Kingdom of Great Britain and Northern Ireland, and Republic of Ireland this EULA shall be governed by and construed in accordance with the laws of England and Wales and the parties accept the exclusive jurisdiction of the courts of England and Wales.

13.6 If the SOFTWARE was obtained in Russia, Belarus, Kazakhstan, or any other country of the CIS, except for Ukraine and Moldova, or if the SOFTWARE was obtained (purchased) in Georgia, Latvia, Lithuania or Estonia, this EULA shall be governed by and construed in accordance with the substantive law in force in the Russian Federation.

13.7 If the SOFTWARE was obtained in Albania, Bosnia and Herzegovina, Bulgaria, Croatia, the Czech Republic, Hungary, Israel, Macedonia, Poland, Romania, Slovakia, Slovenia, Turkey, Serbia, Montenegro, Ukraine or Moldova, this EULA shall be governed by and construed in accordance with substantive law of Ukraine, unless the SOFTWARE has been acquired by the person that has a status of consumer according to the Polish Civil Code. Polish law shall apply to such consumers.

13.8 If article 13.7 is applied and You are an entity or a sole proprietor (a person who is in business by himself/herself without the use of a company structure or partners and bears alone full responsibility for the actions of the business), any and all disputes, controversies or differences in opinion arising out of or relating to the EULA shall be finally resolved through arbitration in accordance with the arbitration rules and procedures of the International Commercial Arbitration Court at the Ukrainian Chamber of Commerce and Industry according to its Order. Judgment of the Court mentioned above is final and obligatory for execution by both parties. If article 13.7 is applied
and You are an individual person, the Shevchenkovsky District Court of Kiev, Ukraine, shall have exclusive jurisdiction over all disputes relating to this EULA.

13.9 If article 13.6 is applied and You are an entity or a sole proprietor (a person who is in business by himself/herself without the use of a company structure or partners and bears alone full responsibility for the actions of the business) the Arbitrazh Court of Moscow, the Russian Federation, shall have exclusive jurisdiction over all disputes relating to this EULA. If article 13.6 is applied and You are an individual person, the Kuzminsky District Court of Moscow, the Russian Federation, shall have exclusive jurisdiction over all disputes relating to this EULA.

13.10 In the cases described in articles 13.1 -13.7, this EULA will not be governed by the conflict of law rules of any jurisdiction or the United Nations Convention on Contracts for the International Sale of Goods, the application of which is expressly excluded.

13.11 If the SOFTWARE was obtained (purchased) in a country other than the countries specified in articles 13.1 -13.7, this EULA shall be governed by and construed in accordance with the substantive law of the country in which You obtained (purchased) the SOFTWARE.

14. Termination

14.1 Unless otherwise agreed by You and ABBYY or an ABBYY Partner in a separate written agreement or except as otherwise provided by the EULA or the documentation for the SOFTWARE, this EULA is effective in perpetuity from the date which You first demonstrate Your acceptance as provided at the beginning of the EULA or as long as it is permitted under applicable law. To the extent that applicable law requires the statement of an expiration period for this EULA, this EULA will last as long as permitted, but in any event, at least as long as the duration of the SOFTWARE copyright and shall automatically expire without further notice when such time period elapses.

14.2 Without prejudice to any other rights, ABBYY may terminate this EULA if You fail to comply with the terms and conditions of this EULA. In such an event, You must destroy all copies of the SOFTWARE, all of its component parts and remove the SOFTWARE from your Computers.

14.3 You may terminate this EULA by destroying all copies of the SOFTWARE, all of its component parts and removing the SOFTWARE.

14.4 Such termination does not relieve You of Your obligation to pay for the SOFTWARE. Articles 2, 3.3, 3.4, 7, 8, 9, 10, 11, 12, 13, 14, 15 and articles 3.8.1, 3.8.4 and 3.7.4 survive the termination or expiration of this EULA, howsoever caused, but this shall not imply or create any continued right to use the SOFTWARE after termination or expiration of this EULA.

15. Miscellaneous

15.1 ABBYY may provide You with printed materials, including the user’s guide or other SOFTWARE documentation.

15.2 In the course of SOFTWARE activation, installation, operation and/or technical support and
maintenance of the SOFTWARE You may be required and You hereby agree to provide ABBYY with certain personal information of Yours (such as but not limited to Your name, address, email address, phone number). You may choose not to provide ABBYY with your personal information, in which case You may not be able to access some or all of the features of ABBYY’s SOFTWARE or obtain any support or other services otherwise available to ABBYY customers who provided their personal information, if provision of personal information is essential to provide You with technical support or maintenance of the SOFTWARE or is requirement of such technical support or maintenance of the SOFTWARE and it does not contradict applicable law You agree not to provide any more personal information than ABBYY or ABBYY’s Partner require, and agree that Your personal information may be processed (including but not limited to collected and/or otherwise used) by ABBYY and/or its affiliates or ABBYY Partners in compliance with applicable law provided that the confidentiality of the data and data security is maintained if it is required by applicable law. Any personal information You provide to ABBYY or its affiliates or ABBYY Partners will be processed, stored and used strictly within ABBYY and its affiliates or ABBYY Partners and will not be disclosed to any third party, except as may be required by applicable law, and, in case of ABBYY USA Software House Inc., will be maintained in accordance with its privacy policy found at http://abbyy.com/privacy. All personal records will be processed solely for the purposes of fulfillment obligations under this EULA by ABBYY. If article 13.7. applies Your personal information will be processed in a database named “Physical persons whose personal information is processed during business activity”.

15.3 You agree that the SOFTWARE may require connections via the Internet or via e-mail with an ABBYY server during SOFTWARE Activation, deactivation or update. ABBYY will take reasonable steps to insure that no personal information about You is transferred during such connections.

15.4 ABBYY may send You e-mails containing product and company news, information about special offers, tips on using ABBYY products and other product and company-related information provided that You agree to receive such information. You may remove Your e-mail address from ABBYY’s mailing list at any time.

15.5 If any claims or lawsuits are brought against You in connection with Your use of the SOFTWARE, You shall inform ABBYY in writing about them promptly and whenever practicable within three (3) days from the moment You learn of them. You shall take all necessary actions to enable ABBYY to take part in the proceedings or hearings of or take over the defense of said claims or lawsuits in court or arbitration, and shall provide ABBYY with Your full cooperation and all information ABBYY deems useful or necessary for the defense of settlement of the corresponding claims or lawsuits, promptly upon (and whenever practicable no later than seven (7) days from the moment of) the receipt of an inquiry from ABBYY.

15.6 Unless otherwise agreed by the parties, the SOFTWARE licensed under this EULA requires the payment of a consideration. Consideration under this EULA is the price of the License established by ABBYY or an ABBYY Partner and payable in accordance with the payment procedures established by them, or may be included in value of equipment or hardware obtained by You, or is part of the consideration payable by You for the full version of the SOFTWARE.

15.7 If any part of this EULA is found void or unenforceable, it will not affect the validity of the balance of the EULA, which shall remain valid and enforceable according to its terms.
ABBYY Recognition Server 4 COM-based API
CreateXmlTicket Method of Client Object

This method creates an XmlTicket object based on the specified workflow.

Visual Basic Syntax

Method CreateXmlTicket(
    workflowName As String
) As XmlTicket

C++ Syntax

HRESULT CreateXmlTicket(
    BSTR workflowName
    IXmlTicket** result
);

Parameters

workflowName
[in] This parameter contains the workflow name. The workflow name must be an element of the list returned by the IClient::Workflows property.

result
[out] A pointer to an IXmlTicket* pointer variable that receives the interface pointer of the XmlTicket object.

Return Values

This method has no specific return values. It returns standard return values of ABBYY Recognition Server Open API functions.

See also

Client, XmlTicket.

See sample: Sample for ASP.NET.
ABBY Recognition Server 4 COM-based API
CreateOutputFormatSettings Method of Client Object

This method creates an **OutputFormatSettings** object.

**Visual Basic Syntax**

```vbnet
Method CreateOutputFormatSettings(
    format As OutputFileFormatEnum
) OutputFormatSettings
```

**C++ Syntax**

```cpp
HRESULT CreateOutputFormatSettings(
    OutputFileFormatEnum format
    IOutputFormatSettings** result
);
```

**Parameters**

- **format**
  - [in] This variable specifies the format of the output file. See the **OutputFileFormatEnum** description for the supported file formats.

- **result**
  - [out] A pointer to an **IOutputFormatSettings*** pointer variable that receives the interface pointer of the **OutputFormatSettings** object.

**Return Values**

This method has no specific return values. It returns standard return values of ABBYY Recognition Server Open API functions.

**See also**

- **Client**
- **OutputFormatSettings**
ABBYY Recognition Server 4 COM-based API
Connect Method of Client Object

This method establishes a connection with the server.

If the server is inaccessible, the method creates an exception: COMException.

If a connection is established with a server that has been stopped, the following text is displayed in the exception: "ABBYY Recognition Server is not available: The client has successfully connected to the server, but the server is not running."

If there is no connection with the server, the following text is displayed in the exception: "ABBYY Recognition Server is not available: Unable to perform RPC call. The call recipient process is not started or its host is unavailable."

Visual Basic Syntax

```vbnet
Method Connect(
    name As String
)
```

C++ Syntax

```c++
HRESULT Connect(
    BSTR name
);
```

Parameters

name

[in] This parameter contains the DNS name of the server or its IP address.

Return Values

This method has no specific return values. It returns standard return values of ABBYY Recognition Server Open API functions.

See also

Client.

See samples: Hello, Listening, AsyncProcessing, Sample for ASP.NET.
ABBYY Recognition Server 4 COM-based API
CreateDocumentAttribute Method of Client Object

This method creates a DocumentAttribute object.

Visual Basic Syntax

Method CreateDocumentAttribute(
    Type As AttributeTypeEnum
) DocumentAttribute

C++ Syntax

HRESULT CreateDocumentAttribute(
    AttributeTypeEnum Type,
    IDocumentAttribute** Result
);

Parameters

Type
[in] This variable specifies the type of the document attribute. See the AttributeTypeEnum description for the supported file formats.

Result
[out] A pointer to an IDocumentAttribute* pointer variable that receives the interface pointer of the DocumentAttribute object.

Return Values

This method has no specific return values. It returns standard return values of ABBYY Recognition Server Open API functions.

See also

Client, DocumentAttribute
ABBY Recognition Server 4 COM-based API
CreateInputFile Method of Client Object

This method creates a **InputFile** object.

**Visual Basic Syntax**

Method `CreateInputFile()

**C++ Syntax**

`HRESULT CreateInputFile(
    IInputFile** Result
);`

**Parameters**

*Result*

[out] A pointer to an **IInputFile** pointer variable that receives the interface pointer of the **InputFile** object.

**Return Values**

This method returns an InputFile object.

**See also**

Client, InputFile
ABBYY Recognition Server 4 COM-based API
DeleteJob Method of IClient Interface

Visual Basic Syntax

Method DeleteJob(
    jobId As String,
)

C++ Syntax

HRESULT DeleteJob(
    BSTR jobId,
);

Parameters

jobId
    [in] This variable contains the job ID.

Return Values

This method has no specific return values.

See also

IClientEvents, Client.
IsListening Method of Client Object

This method allows you to learn whether receipt of job completion notifications from a selected workflow is enabled or disabled. The **StartListening** starts or stops the receipt of notifications. Between the "start receipt" call and the "stop receipt" call of the **StartListening** method, the **IsListening** method returns TRUE.

**Visual Basic Syntax**

Method **IsListening**(
  workflow As String
) Boolean

**C++ Syntax**

```cpp
HRESULT IsListening(
  BSTR workflow
  VARIANT_BOOL* result
);
```

**Parameters**

*workflow*
   - [in] This parameter contains the workflow name. The workflow name must be an element of the list returned by the **IClient::Workflows** property.

*result*
   - [out] A pointer to a Boolean variable that specifies whether receipt of job completion notifications from a selected workflow is enabled or disabled. Must not be NULL.

**Return Values**

This method has no specific return values. It returns standard return values of ABBYY Recognition Server Open API functions.

**See also**

**Client**,  
**IClient::StartListening**.

See sample: **Listening**.
ABBYY Recognition Server 4 COM-based API
GetJobState Method of Client Object

This method allows you to get information about current state of the specified job in asynchronous mode.

Visual Basic Syntax

Method GetJobState(
    JobId As String,
    ByRef State As JobStateEnum
    ByRef Progress As Long
)

C++ Syntax

HRESULT GetJobState(
    BSTR JobId,
    JobStateEnum* State
    Long* Progress
);

Parameters

JobId
   [in] This parameter contains the job ID. The job ID must be a string returned by the IClient::ProcessFileAsync or IClient::ProcessXMLTicketAsync method.

State
   [out] A pointer to a JobStateEnum variable that specifies current state of the job. Must not be NULL.

Progress
   [out] A pointer to a Long variable that specifies the job progress in a percent format. Must not be NULL.

Return Values

This method has no specific return values. It returns standard return values of ABBYY Recognition Server Open API functions.

See also

Client,
JobStateEnum.
ABBY Recognition Server 4 COM-based API
ProcessFile Method of Client Object

This method uses the parameters of the specified workflow to recognize the specified input image.

Visual Basic Syntax

Method ProcessFile(
    fileName As String,
    workflowName As String,
    password As String
) As XmlResult

C++ Syntax

HRESULT ProcessFile(
    BSTR fileName,
    BSTR workflowName,
    BSTR password,
    IXmlResult** result
);

Parameters

fileName
    [in] This parameter contains the image file name.

workflowName
    [in] This parameter contains the workflow name. The workflow name must be an element of the list returned by the IClient::Workflows property.

password
    [in] This parameter contains the password for accessing the PDF file. If a non-PDF file is being recognized, this parameter is ignored. The default value is empty string.

result
    [out] A pointer to an IXmlResult* pointer variable that receives the interface pointer of the XmlResult object.

Return Values

This method has no specific return values. It returns standard return values of ABBYY Recognition Server Open API functions.

See also

Client,
IClient::ProcessFileAsync.
See sample: Hello.
**ProcessFileAsync Method of Client Object**

This method uses the parameters of the specified workflow to recognize the input image in asynchronous mode.

**Visual Basic Syntax**

```vbnet
Method ProcessFileAsync(
    fileName As String,
    workflowName As String,
    password As String
) As String
```

**C++ Syntax**

```cpp
HRESULT ProcessFileAsync(
    BSTR fileName,
    BSTR workflowName,
    BSTR password,
    BSTR* jobId
);
```

**Parameters**

- **fileName**
  
  [in] This parameter contains the image file name.

- **workflowName**
  
  [in] This parameter contains the workflow name. The workflow name must be an element of the list returned by the **IClient::Workflows** property.

- **password**
  
  [in] This parameter contains the password for accessing the PDF file. If a non-PDF file is being recognized, this parameter is ignored. The default value is an empty string.

- **jobId**
  
  [out] A pointer to a string variable that receives the job identification.

**Return Values**

This method has no specific return values. It returns standard return values of ABBYY Recognition Server Open API functions.

**See also**

[Client](#), **IClient::ProcessFile**.
See sample: AsyncProcessing.
ABBYY Recognition Server 4 COM-based API
ProcessXmlTicket Method of Client Object

This method uses the parameters of Xml Ticket and the specified workflow to recognize the input image. The Xml Ticket parameters override the workflow parameters.

**Visual Basic Syntax**

```vbnet
Method ProcessXmlTicket(
    ticket As XmlTicket,
    workflowName As String
) As XmlResult
```

**C++ Syntax**

```cpp
HRESULT ProcessXmlTicket(
    XmlTicket* ticket,
    BSTR workflowName,
    IXmlResult** result
);
```

**Parameters**

- `ticket`  
  [in] This variable refers to the `XmlTicket` object that corresponds to the processing parameters of one job.

- `workflowName`  
  [in] This parameter contains the workflow name. The workflow name must be an element of the list returned by the `IClient::Workflows` property.

- `result`  
  [out] A pointer to an `IXmlResult*` pointer variable that receives the interface pointer of the `XmlResult` object.

**Return Values**

This method has no specific return values. It returns standard return values of ABBYY Recognition Server Open API functions.

**See also**

- `Client`
- `IClient::ProcessXmlTicketAsync`

**See sample:** Sample for ASP.NET.
ABBYY Recognition Server 4 COM-based API
ProcessXmlTicketAsync Method of Client Object

This method uses the parameters of Xml Ticket and the specified workflow to recognize the input image in asynchronous mode. The Xml Ticket parameters override the workflow parameters.

Visual Basic Syntax

Method ProcessXmlTicketAsync(
    ticket As XmlTicket,
    workflowName As String
) As String

C++ Syntax

HRESULT ProcessXmlTicketAsync(
    XmlTicket* ticket,
    BSTR workflowName,
    BSTR* jobId
);

Parameters

ticket
   [in] This variable refers to the XmlTicket object that corresponds to the processing parameters of one job.

workflowName
   [in] This parameter contains the workflow name. The workflow name must be an element of the list returned by the IClient::Workflows property.

jobId
   [out] A pointer to a string variable that receives the job identification.

Return Values

This method has no specific return values. It returns standard return values of ABBYY Recognition Server Open API functions.

See also

Client, IClient::ProcessXmlTicket
ABBYY Recognition Server 4 COM-based API
StartListening Method of Client Object

This method allows you to get notifications about the completion of work by a specified workflow. The `doStart` parameter starts or stops the receipt of notifications. If the method is repeatedly called for the same workflow with the same value of the `doStart` parameter (twice "start receipt" or twice "stop receipt"), redundant calls are ignored. You will receive notifications about jobs which are completed between the "start receipt" call and the "stop receipt" call and whose resulting XML files are saved into the Output folder. If you want the resulting XML file to be saved into the Output folder, you must select the **Save XML Result file in folder** option on the **Output** tab of the **Workflow Properties** dialog box.

The **IsListening** method allows you to learn whether receipt of notifications from a selected workflow is enabled or disabled.

**Visual Basic Syntax**

```vbnet
Method StartListening(
    workflow As String,
    doStart As Boolean
)
```

**C++ Syntax**

```cpp
HRESULT StartListening(
    BSTR workflow,
    VARIANT_BOOL doStart
);
```

**Parameters**

- **workflow**
  
  [in] This parameter contains the workflow name. The workflow name must be an element of the list returned by the **IClient::Workflows** property.

- **doStart**
  
  [in] This variable of the Boolean type starts or stops the receipt of notifications.

**Return Values**

This method has no specific return values. It returns standard return values of ABBYY Recognition Server Open API functions.

**See also**

- **Client**
IClient::IsListening.

See sample: Listening.
ABYY Recognition Server 4 COM-based API
UpdateWorkflow Method of Client Object

This method allows you to change settings of the specified workflow.

**Visual Basic Syntax**

```vbnet
Method GetJobState(
    ByRef workflow As IWorkflow
)
```

**C++ Syntax**

```cpp
HRESULT UpdateWorkflow(
    IWorkflow* workflow
);
```

**Parameters**

- **workflow**
  - [in] A pointer to a `Workflow` variable that represents new settings of the workflow.

**Return Values**

This method has no specific return values. It returns standard return values of ABBYY Recognition Server Open API functions.

**See also**

- `Workflow`.
ABBY Recognition Server 4 COM-based API
OnJobComplete Method of IClientEvents Interface

This method is implemented on the client side. It is called by the ABBYY Recognition Server COM-based API when some of the methods of the Client object are running. It provides information about completion of the job with the specified ID to the client.

Job ID can be changed during job processing, e.g. if document separation is turned on in the workflow settings. In this case, to receive notifications about job completion, you should first start the receipt of notifications from the corresponding workflow with the help of the StartListening method of the Client object. This workflow must publish XML Result files for processed jobs to the Output folder.

Visual Basic Syntax

```
Sub OnJobComplete(
    ByVal jobId As String,
    ByVal result As XmlResult
)
```

C++ Syntax

```
HRESULT OnJobComplete(
    BSTR jobId,
    IXmlResult* result
);
```

Parameters

`jobId`

[in] This variable contains the job ID.

`result`

[in] This variable refers to the XmlResult object that contains the parameters and results of processing one job.

Remarks

The client implementation of this method must assure that all exceptions thrown inside the method are caught and handled and no exceptions are propagated outside the method. Propagation of an exception outside the method may lead to unpredictable results (such as program termination).

See also
IClientEvents,
Client.

See samples: Listening, AsyncProcessing.
ABBYY Recognition Server 4 COM-based API
AddImage Method of XmlTicket Object

This method adds an image to XmlTicket.

**Visual Basic Syntax**

```vbnet
Method AddImage(
    fileName As String
    password As String
)
```

**C++ Syntax**

```csharp
HRESULT AddImage(
    BSTR fileName
    BSTR password
);
```

**Parameters**

- `fileName`  
  [in] This parameter contains the file name.
- `password`  
  [in] This parameter contains the password for accessing the PDF file. The default value is an empty string.

**Return Values**

This method has no specific return values. It returns standard return values of ABBYY Recognition Server Open API functions.

**See also**

- [XmlTicket](#)

**See sample:** [Sample for ASP.NET](#)
LoadFromFile Method

This method restores the contents of the object from a file on disk.

Visual Basic Syntax

Method LoadFromFile(
    path As String
)

C++ Syntax

HRESULT LoadFromFile(
    BSTR path
);

Parameters

path
[in] A path to the file on disk where the contents of the object is stored. The file specified in this path must contain a valid XML representation of an object of the same type as the current one. For example, the file can be obtained as a result of a call to the SaveToFile method of the same type object.

Return Values

This method has no specific return values. It returns standard return values of ABBYY Recognition Server Open API functions.

Remarks

The following objects supports this method: XmlTicket, XmlResult.

See also

SaveToFile
ABBYY Recognition Server 4 COM-based API
SaveToFile Method

This method saves the contents of the object into a file on disk.

Visual Basic Syntax

Method SaveToFile(
    path As String
)

C++ Syntax

HRESULT SaveToFile(
    BSTR path
);

Parameters

path

[in] This parameter specifies the path to the file where the contents of the object should be saved. If a file with this name already exists, it will be overwritten without prompt.

Return Values

This method has no specific return values. It returns standard return values of ABBYY Recognition Server Open API functions.

Remarks

The following objects supports this method: XmlTicket, XmlResult.

The contents of the XmlResult object are saved into the path which is specified in the path parameter if a folder for XML Result file is not specified in the Remote Administration Console (on the Output tab of the Workflow Properties dialog box). If a folder is specified in Remote Administration Console, an XML file will be saved to this folder and the path parameter will be ignored.

See also

LoadFromFile
ABBYY Recognition Server 4 COM-based API
Count Property of Collection Objects

This property stores the number of elements in an ABBYY Recognition Server Open API collection. Each ABBYY Recognition Server collection object has this property.

Visual Basic Syntax

Property Count As Long
   read-only

C++ Syntax

HRESULT Count(
   long* pVal
);

Parameters

pVal
   [out] A pointer to a long variable that receives the value of this property. Must not be NULL.

Return Values

This function has no specific return values. It returns standard return values of ABBYY Recognition Server Open API functions.

Remarks

The following objects have this property: StringsCollection, OutputFormatSettingsCollection, InputFiles, OutputDocuments, Workflows, DocumentInfoItems, DocumentAttributes.

See also

Item, Working with Collections.

See samples: Hello, Listening, AsyncProcessing, Sample for ASP.NET.
ABBYY Recognition Server 4 COM-based API
Add Method of Collection Object

This method adds a new element at the end of the collection.

Visual Basic Syntax

Method Add(
    item As <CollectionType>
)

C++ Syntax

HRESULT Add(
    <CollectionType> item
);

Parameters

item

[in] This parameter contains the newly added element. Its type depends on the type of collection and is described in the following table:

<table>
<thead>
<tr>
<th>Collection type</th>
<th>Element type (Visual basic/C++)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DocumentAttributes</td>
<td>DocumentAttribute/IDocumentAttribute*</td>
</tr>
<tr>
<td>DocumentTypes</td>
<td>DocumentType/IDocumentType*</td>
</tr>
<tr>
<td>IndexingFields</td>
<td>IndexingField/IIndexingField*</td>
</tr>
<tr>
<td>InputFiles</td>
<td>InputFile/IInputFile*</td>
</tr>
<tr>
<td>OutputFormatSettingsCollection</td>
<td>OutputFormatSettings/IOutputFormatSettings*</td>
</tr>
<tr>
<td>StringsCollection</td>
<td>String/BSTR</td>
</tr>
</tbody>
</table>

Return Values

This method has no specific return values. It returns standard return values of ABBYY Recognition Server COM-based API functions.

See also

Working with Collections.

See sample: Sample for ASP.NET.
ABBYY Recognition Server 4 COM-based API
**Item Method of Collection Objects**

This method provides access to a single element of the collection.

**Visual Basic Syntax**

```vbnet
Method Item(
    index As Long,
  ) As ObjectType
```

**C++ Syntax**

```c
HRESULT Item(
    long index,
    InterfaceType** pVal
);
```

**Parameters**

- **index**
  
  [in] This variable contains the index of the element that is accessed via this method. It must be in the range from 0 to the Number of elements - 1, where the number of elements may be received from the Count property of the same collection.

- **ObjectType**
  
  [out] The type of objects in the collection. For example, for the InputFiles collection this type is InputFile.

- **pVal**
  
  [out] A variable of type InterfaceType* that receives a pointer to the interface of the collection element. pVal must not be NULL. *pVal is guaranteed to be non-NULL after a successful method call. InterfaceType is the type of the interface of the objects forming the collection.

**Return Values**

This method has no specific return values. It returns standard return values of ABBYY Recognition Server Open API functions.

**Remarks**

The following collections supports this method: StringsCollection, OutputFormatSettingsCollection, InputFiles, OutputDocuments, Workflows, DocumentInfoItems, DocumentAttributes, DocumentTypes, IndexingFields.

**See also**
Working with Collections.

See samples: Hello, Listening, AsyncProcessing, Sample for ASP.NET.
Remove Method of Collection Objects

This method removes an element from a collection by its index.

**Visual Basic Syntax**

```vbnet
Method Remove(
    index As Long
)
```

**C++ Syntax**

```cpp
HRESULT Remove(
    long index
);
```

**Parameters**

`index`

[in] This variable contains the index of the collection element. It should be in the range from 0 to the value of the `Count` property of this collection minus 1.

**Return Values**

This method has no specific return values. It returns standard return values of ABBYY Recognition Server Open API functions.

**Remarks**

The following collections supports this method: `StringsCollection`, `OutputFormatSettingsCollection`, `InputFiles`, `DocumentInfoItems`, `DocumentAttributes`, `DocumentTypes`, `IndexingFields`.

**See also**

- `RemoveAll`
- `Add`
- `Working with Collections`
ABBYY Recognition Server 4 COM-based API
RemoveAll Method of Collection Objects

This method removes all the elements from a collection and empties it.

Visual Basic Syntax

Method RemoveAll()

C++ Syntax

HRESULT RemoveAll();

Return Values

This method has no specific return values. It returns standard return values of ABBYY Recognition Server Open API functions.

Remarks

The following collections supports this method: StringsCollection, OutputFormatSettingsCollection, InputFiles, DocumentInfoItems, DocumentAttributes, DocumentTypes, IndexingFields.

See also

Remove, Add, Working with Collections
ABBY Recognition Server 4 COM-based API
Pages Object

This object represents a collection of Page objects in the input image file.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count</td>
<td>Long</td>
<td>read-only Stores the number of elements in the collection.</td>
</tr>
</tbody>
</table>

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Item</td>
<td>Provides access to a single element of the collection.</td>
</tr>
</tbody>
</table>

See also

Page, InputFile, Working with Collections.
ABBYY Recognition Server 4 COM-based API
PagePositions Object

This object represents a collection of PagePosition objects.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count</td>
<td>Long, read-only</td>
<td>Stores the number of elements in the collection.</td>
</tr>
</tbody>
</table>

Methods

Name Description

Item Provides access to a single element of the collection.

See also

ABBYY Recognition Server 4 COM-based API
CreateNew Method of DocumentTypes Object

This method creates a new DocumentType object and returns a reference to it.

**Visual Basic Syntax**

```vbnet
Method CreateNew(
    name As String,
) As DocumentType
```

**C++ Syntax**

```cpp
HRESULT CreateNew(
    BSTR name,
    IDocumentType** item
);
```

**Parameters**

- **name**
  
  [in] This parameter contains the name of new document type.

- **item**
  
  [out] A pointer to the DocumentType object.

**Return Values**

This method has no specific return values. It returns standard return values of ABBYY Recognition Server COM-based API functions.

**See also**

[Working with Collections](#)
CreateNew Method of IndexingFields Object

This method creates a new `IndexingField` object and returns a reference to it.

**Visual Basic Syntax**

```vbnet
Method CreateNew(
    Type As AttributeTypeEnum,
    IsRequired As Boolean,
    name As String,
) As IndexingField
```

**C++ Syntax**

```cpp
HRESULT CreateNew(
    AttributeTypeEnum Type,
    VARIANT_BOOL IsRequired,
    BSTR name,
    IIndexingField** item
);
```

**Parameters**

*Type*  
[in] This parameter specifies the type of indexing field. See the `AttributeTypeEnum` description for the supported fields.

*IsRequired*  
[in] This parameter specifies if indexing field is required.

*name*  
[in] This parameter contains the name of new indexing field.

*item*  
[out] A pointer to the `IndexingField` object.

**Return Values**

This method has no specific return values. It returns standard return values of ABBYY Recognition Server COM-based API functions.

**See also**

[Working with Collections](#)
HeaderAndFooterSettings Object
(IHeaderAndFooterSettings Interface)

This object provides functionality for tuning header and footer parameters when exporting recognized text to PDF (PDF/A) format. These parameters are set in the HeaderAndFooter property of the PDFExportSettings (PDFAExportSettings) object. The HeaderAndFooterSettings object allows you to do the following:

- Place headers and footers on each document page;
- Specify the text to be stamped, the location of the header or footer on the page, and other formatting options;
- Include into the header or footer any static text, as well as page numbers, Bates numbers (auto-incrementing number), current date and time, output file name, etc. To include a variable into the footer or header text, insert a suitable tag from the list below:
  - `<BatesNum>` Auto-incrementing numbers
  - `<Dd>` Current date
  - `<EmailSubject>` Subject of the incoming e-mail message
  - `<FileName>` Name of the output file
  - `<Folder>` Name of parent subfolder in image folder
  - `<Mm>` Current month
  - `<PageNum>` Page number
  - `<Yy>` Current year (2 digits)
  - `<Yyyy>` Current year (4 digits)

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BottomMargin</td>
<td>String</td>
<td>Sets the bottom margin. The value of this property must be in the range of 0 to 120 inches (0 to 3048 millimeters). The default value is 0.5 inches.</td>
</tr>
<tr>
<td>CentralFooter</td>
<td>String</td>
<td>Specifies the central footer text. The default value is an empty string.</td>
</tr>
<tr>
<td>CentralHeader</td>
<td>String</td>
<td>Specifies the central header text. The default value is an empty string.</td>
</tr>
<tr>
<td>FontName</td>
<td>String</td>
<td>Sets the font name. The default value is “Times New Roman”.</td>
</tr>
<tr>
<td>FontSize</td>
<td>String</td>
<td>Sets the font size in points. The default value is 12.</td>
</tr>
<tr>
<td>Property</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>----------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>IsBold</td>
<td>String</td>
<td>Specifies if the bold font style has to be used. The default value is FALSE.</td>
</tr>
<tr>
<td>IsInInches</td>
<td>String</td>
<td>Specifies whether measurement units for the margins are inches. If the value of this property is FALSE, the measurement units for the margins are millimeters. The default value is TRUE.</td>
</tr>
<tr>
<td>IsItalic</td>
<td>Long</td>
<td>Specifies if the italic font style has to be used. The default value is FALSE.</td>
</tr>
<tr>
<td>IsUnderlined</td>
<td>Boolean</td>
<td>Specifies if the underlined font style has to be used. The default value is FALSE.</td>
</tr>
<tr>
<td>LeftFooter</td>
<td>Boolean</td>
<td>Specifies the left footer text. The default value is an empty string.</td>
</tr>
<tr>
<td>LeftHeader</td>
<td>Boolean</td>
<td>Specifies the left header text. The default value is an empty string.</td>
</tr>
<tr>
<td>LeftMargin</td>
<td>Long</td>
<td>Sets the left margin. The value of this property must be in the range of 0 to 120 inches (0 to 3048 millimeters). The default value is 1 inch.</td>
</tr>
<tr>
<td>NumberOfDigits</td>
<td>Double</td>
<td>Sets the number of digits in the Bates numbers. If the number of digits in a Bates number is less than specified by this property, the corresponding number of zeros will be added in front of the Bates number. The maximal allowed number of digits is 9. The default value is 5.</td>
</tr>
<tr>
<td>RightFooter</td>
<td>Double</td>
<td>Specifies the right footer text. The default value is an empty string.</td>
</tr>
<tr>
<td>RightHeader</td>
<td>Double</td>
<td>Specifies the right header text. The default value is an empty string.</td>
</tr>
<tr>
<td>RightMargin</td>
<td>Double</td>
<td>Sets the right margin. The value of this property must be in the range of 0 to 120 inches (0 to 3048 millimeters). The default value is 1 inch.</td>
</tr>
<tr>
<td>StartingNumber</td>
<td>Boolean</td>
<td>Sets the starting Bates number. The number of digits in the starting number must not exceed the number specified in the NumberOfDigits property. The default value is 1.</td>
</tr>
<tr>
<td>TextColor</td>
<td>Long</td>
<td>Sets the text color in RGB format. The default value is red or RGB(255, 0, 0).</td>
</tr>
<tr>
<td>TopMargin</td>
<td>Long</td>
<td>Sets the top margin. The value of this property must be in the range of 0 to 120 inches (0 to 3048 millimeters). The default value is 0,5 inches.</td>
</tr>
</tbody>
</table>

**Note:** The Long value is calculated from the RGB triplet using the formula: \((\text{red value}) + (256 \times \text{green value}) + (65536 \times \text{blue value})\), where red value is the first triplet component, green value is the second triplet component, blue value is the third triplet component. Hence the Long value of the color red equals 255.

See also

- IPDFExportSettings::HeaderAndFooter
- IPDFAExportSettings::HeaderAndFooter
ABBYY Recognition Server 4 COM-based API
FontFormattingModeEnum

FontFormattingModeEnum enumeration constants are used to select the desired character formatting mode: plain, restricted or full.

typedef enum {
    FFM_Full,
    FFM_Plain,
    FFM_Restricted
} FontFormattingModeEnum;

Elements

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FFM_Full</td>
<td>The character formatting mode is <strong>Full</strong>. Produced document maintains the formatting of the original.</td>
</tr>
<tr>
<td>FFM_Plain</td>
<td>The character formatting mode is <strong>Plain</strong>. Text formatting is not preserved.</td>
</tr>
<tr>
<td>FFM_Restricted</td>
<td>The character formatting mode is <strong>Restricted</strong>. Retains fonts, font sizes, and paragraphs, but does not retain the exact locations of the objects on the page or the spacing. The resulting text will be left-aligned.</td>
</tr>
</tbody>
</table>

See also

AltoExportSettings::FontFormattingMode, IEPUBExportSettings::FontFormattingMode
Add method of the DocumentInfoItems Object

This method adds a new element at the end of the DocumentInfoItems collection.

Visual Basic Syntax

Method Add(
    name As String,
    value As String
)

C++ Syntax

HRESULT Add(
    BSTR name,
    BSTR value
);

Parameters

name
    [in] This parameter contains the newly added key name. The key name must not be an empty string and must not repeat any of the other key names of the collection. You must not add key-value pairs with Creator, Producer, ModDate, and CreationDate key names. The value of the Producer key name can be set in the IPDFExportSettings::Producer (IPDFAExportSettings::Producer) property. The value of the Creator, ModDate, and CreationDate key names cannot be changed.

value
    [in] This parameter contains the value of the newly added key.

Return Values

This method has no specific return values. It returns standard return values of ABBYY Recognition Server COM-based API functions.

See also

DocumentInfoItems, DocumentInfoItem.
ABBYY Recognition Server 4 COM-based API
OutputFormatSettingsCollection Object (IOutputFormatSettingsCollection Interface)

This object provides access to a collection of OutputFormatSettings objects.

Properties

Name   Type       Description
       Count, Long, read-only Stores the number of elements in the collection.

Methods

Name       Description
Add        Adds a new element at the end of the collection.
Item       Provides access to a single element of the collection.
Remove     Removes an element from the collection.
RemoveAll  Removes all the elements from the collection.

See also

OutputFormatSettings, ExportParams,
Working with Collections.

See samples: Hello, Listening, AsyncProcessing, Sample for ASP.NET.
ABBYY Recognition Server 4 COM-based API
InputFiles Object

This object represents a collection of InputFile objects.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count</td>
<td>Long</td>
<td>Stores the number of elements in the collection.</td>
</tr>
</tbody>
</table>

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add</td>
<td>Adds a new element at the end of the collection.</td>
</tr>
<tr>
<td>Item</td>
<td>Provides access to a single element of the collection.</td>
</tr>
<tr>
<td>Remove</td>
<td>Removes an element from the collection.</td>
</tr>
<tr>
<td>RemoveAll</td>
<td>Removes all the elements from the collection.</td>
</tr>
</tbody>
</table>

See also

InputFile, XmlTicket, XmlResult, Working with Collections.

See sample: Sample for ASP.NET.
ABBYY Recognition Server 4 COM-based API
Workflows Object (IWorkflows Interface)

This object provides access to a collection of Workflow objects which represent the workflow settings. A reference to this object is returned by the IClient::WorkflowsSettings property.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count</td>
<td>Long</td>
<td>read-only Stores the number of elements in the collection.</td>
</tr>
</tbody>
</table>

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Item</td>
<td>Provides access to a single element of the collection.</td>
</tr>
</tbody>
</table>

See also

Working with Collections, Workflow, IClient::WorkflowsSettings
Workflow Object (IWorkflow Interface)

This object represents the workflow settings: the workflow name, the paths to the Input folder and the Exception folder, and the export parameters. All the properties are read-only.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ExceptionsFolder</td>
<td>String, read-only</td>
<td>Returns the path to the Exceptions Folder of the workflow.</td>
</tr>
<tr>
<td>ExportParams</td>
<td>ExportParams, read-only</td>
<td>Returns the export parameters.</td>
</tr>
<tr>
<td>IndexingSettings</td>
<td>IndexingSettings, read-only</td>
<td>Returns the indexing settings.</td>
</tr>
<tr>
<td>InputFolder</td>
<td>String, read-only</td>
<td>Returns the path to the Input Folder of the workflow.</td>
</tr>
<tr>
<td>Name</td>
<td>String, read-only</td>
<td>Returns the name of the workflow.</td>
</tr>
</tbody>
</table>

See also

Workflows
ABBYY Recognition Server 4 COM-based API
DocumentInfoItems Object (IDocumentInfoItems Interface)

This object provides access to a collection of DocumentInfoItem objects which represent metadata from the PDF (PDF/A) file. A reference to this object is returned by the IPDFExportSettings::DocumentInfo (IPDFAExportSettings::DocumentInfo) property.

You can add key-value pairs to the DocumentInfoItems collection using the Add method. The key name (IDocumentInfoItem::Name) must not be an empty string and must not repeat any of the other key names of the collection.

You must not add key-value pairs with Creator and Producer key names. The value of the Producer key name can be set in the IPDFExportSettings::Producer (IPDFAExportSettings::Producer) property. The value of the Creator key name cannot be changed. It is the same as in the source document.

You must not add key-value pairs with ModDate and CreationDate key names. The value of the ModDate and CreationDate key names cannot be changed. The value of the ModDate key is set to the current date, the value of the CreationDate key is copied from the source document, or is set to the current date.

If you set the values of the Author, Keywords, Subject or Title keys, the values of the corresponding properties of the PDFExportSettings (PDFAExportSettings) object will be set to the same value. Vice versa, if you change the values of the Author, Keywords, Subject or Title properties of the PDFExportSettings (PDFAExportSettings) object, the values of the corresponding keys will be changed or the corresponding key-value pairs will be added to the collection.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count</td>
<td>Long, read-only</td>
<td>Stores the number of elements in the collection.</td>
</tr>
</tbody>
</table>

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add</td>
<td>Adds a new DocumentInfoItem object at the end of the collection.</td>
</tr>
<tr>
<td>Item</td>
<td>Provides access to a single element of the collection.</td>
</tr>
<tr>
<td>Remove</td>
<td>Removes an element from the collection.</td>
</tr>
<tr>
<td>RemoveAll</td>
<td>Removes all the elements from the collection.</td>
</tr>
</tbody>
</table>
See also

Working with Collections,
DocumentInfoItem,
IPDFExportSettings::DocumentInfo,
IPDFAExportSettings::DocumentInfo
ABBYY Recognition Server 4 COM-based API
DocumentTypes Object

This object represents a collection of DocumentType objects.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count</td>
<td>Long,</td>
<td>Stores the number of elements in the collection.</td>
</tr>
</tbody>
</table>

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add</td>
<td>Adds a new element at the end of the collection.</td>
</tr>
<tr>
<td>CreateNew</td>
<td>Creates a new DocumentType object and returns a reference to it.</td>
</tr>
<tr>
<td>Item</td>
<td>Provides access to a single element of the collection.</td>
</tr>
<tr>
<td>Remove</td>
<td>Removes an element from the collection.</td>
</tr>
<tr>
<td>RemoveAll</td>
<td>Removes all the elements from the collection.</td>
</tr>
</tbody>
</table>

See also

DocumentType,
Working with Collections.
ABBYY Recognition Server 4 COM-based API
DocumentType Object (IDocumentType Interface)

This object represents document type for indexing.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>String, read-only</td>
<td>Returns the name of the document type.</td>
</tr>
<tr>
<td>Fields</td>
<td>IndexingFields,</td>
<td>read-only Returns a collection of indexing fields.</td>
</tr>
</tbody>
</table>

See also

IndexingFields.
ABYY Recognition Server 4 Web Services API
IndexingFields Object

This object represents a collection of IndexingField objects.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count</td>
<td>Long</td>
<td>Stores the number of elements in the collection.</td>
</tr>
</tbody>
</table>

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add</td>
<td>Adds a new element at the end of the collection.</td>
</tr>
<tr>
<td>CreateNew</td>
<td>Creates a new IndexingField object and returns a reference to it.</td>
</tr>
<tr>
<td>Item</td>
<td>Provides access to a single element of the collection.</td>
</tr>
<tr>
<td>Remove</td>
<td>Removes an element from the collection.</td>
</tr>
<tr>
<td>RemoveAll</td>
<td>Removes all the elements from the collection.</td>
</tr>
</tbody>
</table>

See also

IndexingField, Working with Collections.
IndexingField Object (IIndexingField Interface)

This object represents an indexing field. It is a common interface for interfaces of indexing fields of different types:

- BooleanIndexingField
- EnumerationIndexingField
- MultipleLinesIndexingField
- RegularExpressionIndexingField
- SingleLineIndexingField

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>String, read-only</td>
<td>Returns the indexing field name.</td>
</tr>
<tr>
<td>Type</td>
<td>AttributeTypeEnum, read-only</td>
<td>Returns the indexing field type.</td>
</tr>
<tr>
<td>IsRequired</td>
<td>Boolean, read-only</td>
<td>If the indexing field is required, this property returns TRUE.</td>
</tr>
</tbody>
</table>

All properties are set when the `IIndexingFields::CreateNew` method is called.

See also

IndexingFields.
ABBYY Recognition Server 4 Web Services API Samples
Sample Hello

This sample shows the basic steps of using the ABBYY Recognition Server Web Services API:

- creating the Client object;
- connecting to the server;
- getting the list of available workflows;
- selecting the workflow whose properties will be used for input image recognition;
- input image recognition;
- checking the recognition results.

The sample uses the following methods and properties:

<table>
<thead>
<tr>
<th>Object</th>
<th>Properties</th>
<th>Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client</td>
<td>Workflows</td>
<td>Connect, ProcessFile</td>
</tr>
<tr>
<td>StringsCollection</td>
<td>Count</td>
<td>Item</td>
</tr>
<tr>
<td>XmlResult</td>
<td>IsFailed</td>
<td></td>
</tr>
</tbody>
</table>

The Hello sample code is intended to be used in the following development tools:

- Visual Basic 6.0
- Visual Basic.Net
- raw C++
- C++ with the Native COM Support
- C#

See also

Description of the ABBYY Recognition Server Open API Samples
ABBYY Recognition Server 4 Web Services API Samples
Sample Listening

This sample shows how to use the `Client::IsListening` and `Client::StartListening` methods of the ABBYY Recognition Server Web Services API. It performs the following:

- creating the `Client` object;
- connecting to the server;
- getting the list of available workflows;
- setting up notifications about the completion of work by a specified workflow.

The sample uses the following methods and properties:

<table>
<thead>
<tr>
<th>Object</th>
<th>Properties</th>
<th>Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client</td>
<td>Workflows</td>
<td>Connect, IsListening, StartListening</td>
</tr>
<tr>
<td>StringsCollection</td>
<td>Count</td>
<td>Item</td>
</tr>
<tr>
<td>IClientEvents</td>
<td></td>
<td>OnJobComplete</td>
</tr>
</tbody>
</table>

The `Listening` sample code exists only for Visual Basic 6.0.

See also

Description of the ABBYY Recognition Server Open API Samples
Sample AsyncProcessing

This sample shows the basic steps of using the ABBYY Recognition Server Web Services API for asynchronous processing. It performs the following:

- creating the Client object;
- connecting to the server;
- getting the list of available workflows;
- setting up notifications about the completion of work by a specified workflow.

The sample uses the following methods and properties:

<table>
<thead>
<tr>
<th>Object</th>
<th>Properties</th>
<th>Methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client</td>
<td>Workflows</td>
<td>Connect, ProcessFileAsync</td>
</tr>
<tr>
<td>StringsCollection</td>
<td>Count</td>
<td>Item</td>
</tr>
<tr>
<td>IClientEvents</td>
<td></td>
<td>OnJobComplete</td>
</tr>
</tbody>
</table>

The AsyncProcessing sample code exists only for Visual Basic 6.0.

See also

Description of the ABBYY Recognition Server Open API Samples
Sample for ASP.NET

In the folder %ALLUSERSPROFILE%\Application Data\ABBYY Recognition Server 4.0\Samples\ASP.NET, you can find a sample of ASP.NET code which allows creating Web applications for online image recognition with the help of the ABBYY Recognition Server COM-based API.

Note: In Microsoft Windows Vista, Windows 7, and Windows 8, the sample ASP.NET code is stored in the folder %PUBLIC%\ABBYY\ABBYY Recognition Server 4.0\Samples\ASP.NET.

To run this sample, ASP.NET must be installed on your computer. ASP.NET is a part of Microsoft .NET Framework. To install Microsoft .NET Framework 2.0, you may run dotnetfx20.exe file from the DotNet folder on the Recognition Server 4 CD and follow the installation program instructions. When installing Microsoft .NET Framework, ASP.NET is copied on the computer but not installed. You should install it manually by running:

C:\WINDOWS\Microsoft.NET\Framework\vX.X.XXX\aspnet_regiis.exe -i (specify the appropriate version instead of vX.X.XXX).

To run the ASP.NET sample application, you must do the following:

1. Install Internet Information Services (IIS) (Start > Settings > Control Panel > Add or Remove Programs > Add/Remove Windows Components).
2. Create the Hello folder in the C:\Inetpub\wwwroot folder.
3. Open the IIS administration console (Start > Settings > Control Panel > Administrative Tools > Internet Information Services).
4. In the Default Web Site node, create the "Hello" virtual folder (Action > New > Virtual Directory). Enter Hello in the Alias field, and enter C:\Inetpub\wwwroot\Hello in the Directory field. Make sure that you select the Read and Run scripts options in the Access Permissions section.
5. Open the dialog box of the properties of the created folder (Action > Properties). On the Documents tab, select the Enable Default Document option and change the default document to "Hello.aspx".
6. Copy the contents of the Samples\ASP.NET\Hello folder to the C:\Inetpub\wwwroot\Hello folder.
7. Make sure that the Internet Guest Account has read\write permissions for the C:\Inetpub\wwwroot\Hello\Upload folder and the Input folder, Output folder and Exceptions folder of the first workflow of the server.
8. Restart IIS (select Action > All Tasks > Restart IIS... in the server node menu).
9. Open the project in Microsoft Visual Studio .NET (select File > Open > Project From Web in the menu).
10. Build and start the project.

Once the application is started, a Web form appears. In the Select images section of this form, specify the image files. In the Specify option section, specify the recognition parameters. When you
click **Start Processing...**, the recognition process starts. When this process is over, you will get a page with the results.

The sample uses the following methods and properties:

<table>
<thead>
<tr>
<th>Object</th>
<th>Properties</th>
<th>Methods</th>
<th>Enumeration constants</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Client</strong></td>
<td>Workflows</td>
<td><strong>Connect</strong>, <strong>CreateXmlTicket</strong>, <strong>ProcessXmlTicket</strong></td>
<td></td>
</tr>
<tr>
<td><strong>XmlTicket</strong></td>
<td>PreprocessingParams,</td>
<td><strong>AddImage</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>RecognitionParams, ExportParams</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>InputFiles</strong></td>
<td>Count</td>
<td><strong>Item</strong></td>
<td></td>
</tr>
<tr>
<td><strong>InputFile</strong></td>
<td>OutputFiles</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>RecognitionParams</strong></td>
<td>Languages</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>PreprocessingParams</strong></td>
<td>RotationType</td>
<td><strong>RotationTypeEnum</strong></td>
<td></td>
</tr>
<tr>
<td><strong>StringsCollection</strong></td>
<td>Count</td>
<td><strong>Add</strong>, <strong>Item</strong></td>
<td></td>
</tr>
<tr>
<td><strong>XmlResult</strong></td>
<td>InputFiles</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
GetWorkflows Method of RSSoapService Object

This method returns a set of available workflows on the Recognition Server with specified location. Only workflows with the following types of Input Folder are available: shared folder, FTP folder.

**C# Syntax**

```csharp
string[] GetWorkflows(
    string serverLocation
);
```

**Parameters**

`serverLocation`

[in] This parameter contains the DNS name of the server or its IP address.

[out] An array of string variables that receives a set of available workflows on the Recognition Server with specified location (`serverLocation`).

**See also**

[RSSoapService](#). See sample: [WebServiceHello](#).
ABBYY Recognition Server 4 Web Services API
ProcessFile Method of RSSoapService Object

This method uses the parameters of the specified workflow to recognize the specified input image.

C# Syntax

```csharp
XmlResult ProcessFile(
    string location,
    string workflowName,
    FileContainer file
);
```

Parameters

*location*

[in] This parameter contains the DNS name of the server or its IP address.

*workflowName*

[in] This parameter contains the workflow name. The workflow name must be an element of the list returned by the *RSSoapService::GetWorkflows* method for the same server.

*file*

[in] This parameter contains the *FileContainer* object that corresponds to the processing file.

[out] An *XmlResult* variable that receives the *XmlResult* object.

See also

*RSSoapService*
CreateTicket Method of RSSoapService Object

This method creates an XmlTicket object based on the specified workflow.

C# Syntax

```csharp
XmlTicket CreateTicket(
    string location,
    string workflowName
);
```

Parameters

*location*

[in] This parameter contains the DNS name of the server or its IP address.

*workflowName*

[in] This parameter contains the workflow name. The workflow name must be an element of the list returned by the RSSoapService::GetWorkflows method for the same server.

[out] An XmlResult variable that receives the XmlResult object.

See also

RSSoapService, XmlTicket. See sample: WebServiceHello.
ProcessTicket Method of RSSoapService Object

This method uses the parameters of the XmlTicket object and the specified workflow to recognize the input image. The XmlTicket parameters override the workflow parameters.

C# Syntax

```
XmlResult ProcessTicket(
    string location,
    string workflowName,
    XmlTicket ticket
);
```

Parameters

- **location**
  - [in] This parameter contains the DNS name of the server or its IP address.

- **workflowName**
  - [in] This parameter contains the workflow name. The workflow name must be an element of the list returned by the `RSSoapService::GetWorkflows` method for the same server.

- **ticket**
  - [in] This variable contains the `XmlTicket` object that corresponds to the processing parameters of one job.

    [out] An `XmlResult` variable that receives the `XmlResult` object.

See also

- `RSSoapService`, `XmlTicket`. See sample: `WebServiceHello`.
ABBYY Recognition Server 4 Web Services API
DeleteJob Method of RSSoapService Object

This method completely deletes the job.

C# Syntax

```csharp
DeleteJob(
    string jobId
);
```

Parameters

`jobId`

[out] A string variable that contains the job identification.

See also

RSSoapService
ABBYY Recognition Server 4 Web Services API
GetJobState Method of RSSoapService Object

This method allows to get information about current state of the specified job.

C# Syntax

```csharp
string[] GetJobState(
    string location,
    string jobId
);
```

Parameters

location
  [in] This parameter contains the DNS name of the server or its IP address.

jobId
  [out] A string variable that contains the job identification.

See also

RSSoapService
StartProcessFile Method of RSSoapService Object

This method uses the parameters of the specified workflow to recognize the input image in asynchronous mode.

C# Syntax

```csharp
XmlResult ProcessFile(
    string location,
    string workflowName,
    Hello.RSSoapService.FileContainer file
);
```

Parameters

- **location**
  - [in] This parameter contains the DNS name of the server or its IP address.

- **workflowName**
  - [in] This parameter contains the workflow name. The workflow name must be an element of the list returned by the `RSSoapService::GetWorkflows` method for the same server.

- **file**
  - [in] This parameter contains the `FileContainer` object that corresponds to the processing file.

  - [out] An `XmlResult` variable that receives the `XmlResult` object.

See also

- RSSoapService
StartProcessTicket Method of RSSoapService Object

This method uses the parameters of the XmlTicket object and the specified workflow to recognize the input image in asynchronous mode

C# Syntax

```csharp
XmlResult ProcessTicket(
    string location,
    string workflowName,
    Hello.RSSoapService.XmlTicket ticket
);
```

Parameters

**location**

[in] This parameter contains the DNS name of the server or its IP address.

**workflowName**

[in] This parameter contains the workflow name. The workflow name must be an element of the list returned by the RSSoapService::GetWorkflows method for the same server.

**ticket**

[in] This variable contains the XmlTicket object that corresponds to the processing parameters of one job.

[out] An XmlResult variable that receives the XmlResult object.

See also

RSSoapService,

XmlTicket
ABBYY Recognition Server 4 Web Services API
Pages Object

This object represents a collection of Page objects in the input image file.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count</td>
<td>Long, read-only</td>
<td>Stores the number of elements in the collection.</td>
</tr>
</tbody>
</table>

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Item</td>
<td>Provides access to a single element of the collection.</td>
</tr>
</tbody>
</table>

See also

Page, InputFile, Working with Collections.
ABBY Recognition Server 4 Web Services API
Count Property of Collection Objects

This property stores the number of elements in an ABBYY Recognition Server Open API collection. Each ABBYY Recognition Server collection object has this property.

Visual Basic Syntax

```vbnet
Property Count As Long
    read-only
```

C++ Syntax

```cpp
HRESULT Count(
    long* pVal
);
```

Parameters

*pVal*

[out] A pointer to a *long* variable that receives the value of this property. Must not be NULL.

Return Values

This function has no specific return values. It returns standard return values of ABBYY Recognition Server Open API functions.

Remarks

The following objects have this property: StringsCollection, OutputFormatSettingsCollection, InputFiles, OutputDocuments, Workflows, DocumentInfoItems, DocumentAttributes.

See also

Item, Working with Collections.

See samples: Hello, Listening, AsyncProcessing, Sample for ASP.NET.
Add Method of Collection Object

This method adds a new element at the end of the collection.

Visual Basic Syntax

Method Add(
    item As `<CollectionType>`
)

C++ Syntax

HRESULT Add(
    `<CollectionType>` item
);

Parameters

`item`
[in] This parameter contains the newly added element. Its type depends on the type of collection and is described in the following table:

<table>
<thead>
<tr>
<th>Collection type</th>
<th>Element type (Visual basic/C++)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DocumentAttributes</td>
<td>DocumentAttribute/IDocumentAttribute*</td>
</tr>
<tr>
<td>DocumentTypes</td>
<td>DocumentType/IDocumentType*</td>
</tr>
<tr>
<td>IndexingFields</td>
<td>IndexingField/IIndexingField*</td>
</tr>
<tr>
<td>InputFiles</td>
<td>InputFile/IInputFile*</td>
</tr>
<tr>
<td>OutputFormatSettingsCollection</td>
<td>OutputFormatSettings/IOutputFormatSettings*</td>
</tr>
<tr>
<td>StringsCollection</td>
<td>String/BSTR</td>
</tr>
</tbody>
</table>

Return Values

This method has no specific return values. It returns standard return values of ABBYY Recognition Server COM-based API functions.

See also

Working with Collections.

See sample: Sample for ASP.NET.
ABBYY Recognition Server 4 Web Services API
Item Method of Collection Objects

This method provides access to a single element of the collection.

**Visual Basic Syntax**

```vbnet
Method Item(
    index As Long,
) As ObjectType
```

**C++ Syntax**

```cpp
HRESULT Item(
    long index,
    InterfaceType** pVal
);
```

**Parameters**

*index*  
[in] This variable contains the index of the element that is accessed via this method. It must be in the range from 0 to the *Number of elements - 1*, where the number of elements may be received from the Count property of the same collection.

*ObjectType*  
[out] The type of objects in the collection. For example, for the InputFiles collection this type is InputFile.

*pVal*  
[out] A variable of type InterfaceType* that receives a pointer to the interface of the collection element. *pVal* must not be NULL. *pVal* is guaranteed to be non-NULL after a successful method call. InterfaceType is the type of the interface of the objects forming the collection.

**Return Values**

This method has no specific return values. It returns standard return values of ABBYY Recognition Server Open API functions.

**Remarks**

The following collections supports this method: StringsCollection, OutputFormatSettingsCollection, InputFiles, OutputDocuments, Workflows, DocumentInfoItems, DocumentAttributes, DocumentTypes, IndexingFields.

See also
Working with Collections.

See samples: Hello, Listening, AsyncProcessing, Sample for ASP.NET.
Remove Method of Collection Objects

This method removes an element from a collection by its index.

Visual Basic Syntax

Method Remove(
    index As Long
)

C++ Syntax

HRESULT Remove(
    long index
);

Parameters

index

    [in] This variable contains the index of the collection element. It should be in the range from 0 to the value of the Count property of this collection minus 1.

Return Values

This method has no specific return values. It returns standard return values of ABBYY Recognition Server Open API functions.

Remarks

The following collections supports this method: StringsCollection, OutputFormatSettingsCollection, InputFiles, DocumentInfoItems, DocumentAttributes, DocumentTypes, IndexingFields.

See also

RemoveAll, Add, Working with Collections
RemoveAll Method of Collection Objects

This method removes all the elements from a collection and empties it.

**Visual Basic Syntax**

Method `RemoveAll()`

**C++ Syntax**

`HRESULT RemoveAll();`

**Return Values**

This method has no specific return values. It returns standard return values of ABBYY Recognition Server Open API functions.

**Remarks**

The following collections supports this method: `StringsCollection`, `OutputFormatSettingsCollection`, `InputFiles`, `DocumentInfoItems`, `DocumentAttributes`, `DocumentTypes`, `IndexingFields`.

**See also**

`Remove`,
`Add`,
`Working with Collections`
PagePositions Object

This object represents a collection of PagePosition objects.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count</td>
<td>Long, read-only</td>
<td>Stores the number of elements in the collection.</td>
</tr>
</tbody>
</table>

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Item</td>
<td>Provides access to a single element of the collection.</td>
</tr>
</tbody>
</table>

See also

PDFAModeEnum

PDFAModeEnum enumeration constants are used to set the PDF/A standard.

```c
enum PDFAModeEnum{
    PDFAM_PdfA_1a,
    PDFAM_PdfA_1b,
    PDFAM_PdfA_2a,
    PDFAM_PdfA_2b,
    PDFAM_PdfA_2u,
    PDFAM_PdfA_3a,
    PDFAM_PdfA_3b,
    PDFAM_PdfA_3u
};
```

Elements

PDF/A-1a — Level A compliance in Part 1 PDF/A-1b — Level B compliance in Part 1 PDF/A-2a — Level A compliance in Part 2 PDF/A-2b — Level B compliance in Part 2 PDF/A-2u — Level B compliance in Part 2 with an additional requirement that all text in the document have Unicode mapping

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PDFAM_PdfA_1a</td>
<td>Level A compliance in Part 1</td>
</tr>
<tr>
<td>PDFAM_PdfA_1b</td>
<td>Level B compliance in Part 1</td>
</tr>
<tr>
<td>PDFAM_PdfA_2a</td>
<td>Level A compliance in Part 2</td>
</tr>
<tr>
<td>PDFAM_PdfA_2b</td>
<td>Level B compliance in Part 2</td>
</tr>
<tr>
<td>PDFAM_PdfA_2u</td>
<td>Level B compliance in Part 2 with an additional requirement that all text in the document have Unicode mapping</td>
</tr>
<tr>
<td>PDFAM_PdfA_3a</td>
<td>Level A compliance in Part 3</td>
</tr>
<tr>
<td>PDFAM_PdfA_3b</td>
<td>Level B compliance in Part 3</td>
</tr>
<tr>
<td>PDFAM_PdfA_3u</td>
<td>Level B compliance in Part 3 with an additional requirement that all text in the document have Unicode mapping</td>
</tr>
</tbody>
</table>

See also

IPDFAExportSettings::Mode
ABBYY Recognition Server 4 Web Services API
Client Object (IClient Interface)

This object allows you to connect to ABBYY Recognition Server and to process recognition jobs in synchronous and asynchronous modes. It can be created by using the CreateObject (Visual Basic) or CoCreateInstance (C++/C) methods.

The Client object is a so-called "connectable object" (see details in the Connectable Objects section). It may be declared WithEvents in Visual Basic. For C++ user this fact means that it supports the IConnectionPointContainer interface. To receive notification events during processing, a C++ user should create an object derived from the IClientEvents interface, then set up the connection between it and events source implemented in the Client object by standard COM means.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Workflows</td>
<td>StringsCollection, read-only</td>
<td>Returns a collection of available workflows. Only workflows with the following types of the Input folder are listed: Shared Folder, FTP Folder.</td>
</tr>
<tr>
<td>WorkflowsSettings</td>
<td>Workflows, read-only</td>
<td>Returns a collection of workflow settings for all available workflows. Only workflows with the following types of the Input folder are listed: Shared Folder, FTP Folder.</td>
</tr>
</tbody>
</table>

Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connect</td>
<td>Establishes a connection with the server.</td>
</tr>
<tr>
<td>CreateDocumentAttribute</td>
<td>Creates a DocumentAttribute object.</td>
</tr>
<tr>
<td>CreateOutputFormatSettings</td>
<td>Creates an OutputFormatSettings object.</td>
</tr>
<tr>
<td>CreateXmlTicket</td>
<td>Creates an XmlTicket object based on the specified workflow.</td>
</tr>
<tr>
<td>IsListening</td>
<td>Allows you to learn whether receipt of job completion notifications from a selected workflow is enabled or disabled.</td>
</tr>
<tr>
<td>GetJobState</td>
<td>Allows you to get information about current state of the specified job.</td>
</tr>
<tr>
<td>ProcessFile</td>
<td>Uses the parameters of the specified workflow to recognize the specified input image.</td>
</tr>
<tr>
<td>ProcessFileAsync</td>
<td>Uses the parameters of the specified workflow to recognize the input image in asynchronous mode.</td>
</tr>
<tr>
<td>ProcessXmlTicket</td>
<td>Uses the parameters of Xml Ticket and the specified workflow to recognize the input image.</td>
</tr>
</tbody>
</table>
ProcessXmlTicketAsync recognize the input image in asynchronous mode.

StartListening Allows you to get notifications about the completion of work by a specified workflow.

UpdateWorkflow Changes settings of the specified workflow

See also

XmlTicket, StringsCollection.

See samples: Hello, Listening, AsyncProcessing, Sample for ASP.NET.
ABBYY Recognition Server 4 COM-based API
ProcessXmlTicketAsync Method of Client Object

This method uses the parameters of Xml Ticket and the specified workflow to recognize the input image in asynchronous mode. The Xml Ticket parameters override the workflow parameters.

Visual Basic Syntax

Method ProcessXmlTicketAsync(
  ticket As XmlTicket,
  workflowName As String
) As String

C++ Syntax

HRESULT ProcessXmlTicketAsync(
  XmlTicket* ticket,
  BSTR workflowName,
  BSTR* jobId
);

Parameters

ticket
  [in] This variable refers to the XmlTicket object that corresponds to the processing parameters of one job.
workflowName
  [in] This parameter contains the workflow name. The workflow name must be an element of the list returned by the IClient::Workflows property.
jobId
  [out] A pointer to a string variable that receives the job identification.

Return Values

This method has no specific return values. It returns standard return values of ABBYY Recognition Server Open API functions.

See also

Client, IClient::ProcessXmlTicket
ABBYY Recognition Server 4 COM-based API
Page Object

This object represents a single page of the input image file.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>String, read-only</td>
<td>Stores the unique identifier of the page. The identifier is generated automatically.</td>
</tr>
<tr>
<td>Number</td>
<td>String, read-only</td>
<td>Stores the page index in the input image file. The indexing starts with 0.</td>
</tr>
</tbody>
</table>

See also

Pages.
ABBYY Recognition Server 4 COM-based API
PagePosition Object

This object stores information about the page position within the input document.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FileId</td>
<td>String, read-only</td>
<td>Stores the unique identifier of the input file. The identifier is generated automatically. The value of this property is the same as the value of InputFile.ID.</td>
</tr>
<tr>
<td>PageId</td>
<td>String, read-only</td>
<td>Stores the unique identifier of the page. The identifier is generated automatically. The value of this property is the same as the value of Page.ID.</td>
</tr>
</tbody>
</table>

See also

PagePositions.
ABBYY Recognition Server 4 COM-based API
IndexingSettings Object (IIndexingSettings Interface)

This object provides functionality for tuning indexing settings.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DefaultDocumentType</td>
<td>String</td>
<td>Specifies the default document type name.</td>
</tr>
<tr>
<td>DocumentTypes</td>
<td>DocumentTypes, read-only</td>
<td>Stores a collection of document types.</td>
</tr>
</tbody>
</table>

See also

DocumentTypes.
ABBY Recognition Server 4 COM-based API
Add method of the DocumentInfoItems Object

This method adds a new element at the end of the DocumentInfoItems collection.

Visual Basic Syntax

```vbnet
Method Add(
    name As String,
    value As String
)
```

C++ Syntax

```c++
HRESULT Add(
    BSTR name,
    BSTR value
);
```

Parameters

**name**

[in] This parameter contains the newly added key name. The key name must not be an empty string and must not repeat any of the other key names of the collection. You must not add key-value pairs with Creator, Producer, ModDate, and CreationDate key names. The value of the Producer key name can be set in the IPDFExportSettings::Producer (IPDFAExportSettings::Producer) property. The value of the Creator, ModDate, and CreationDate key names cannot be changed.

**value**

[in] This parameter contains the value of the newly added key.

Return Values

This method has no specific return values. It returns standard return values of ABBYY Recognition Server COM-based API functions.

See also

DocumentInfoItems, DocumentInfoItem.
ABBY Recognition Server 4 Web Services API
CreateNew Method of DocumentTypes Object

This method creates a new DocumentType object and returns a reference to it.

**Visual Basic Syntax**

Method CreateNew(
    name As String,
) As DocumentType

**C++ Syntax**

HRESULT CreateNew(
    BSTR name,
    IDocumentType** item
);

**Parameters**

- **name**
  [in] This parameter contains the name of new document type.

- **item**
  [out] A pointer to the DocumentType object.

**Return Values**

This method has no specific return values. It returns standard return values of ABBYY Recognition Server COM-based API functions.

**See also**

Working with Collections.
ABBYY Recognition Server 4 COM-based API
CreateNew Method of IndexingFields Object

This method creates a new `IndexingField` object and returns a reference to it.

**Visual Basic Syntax**

Method `CreateNew`

```vbnet
    Type As AttributeTypeEnum,
    IsRequired As Boolean,
    name As String,
) As IndexingField
```

**C++ Syntax**

```cpp
HRESULT CreateNew(
    AttributeTypeEnum Type,
    VARIANT_BOOL IsRequired,
    BSTR name,
    IIndexingField** item
);
```

**Parameters**

*Type*

[in] This parameter specifies the type of indexing field. See the `AttributeTypeEnum` description for the supported fields.

*IsRequired*

[in] This parameter specifies if indexing field is required.

*name*

[in] This parameter contains the name of new indexing field.

*item*

[out] A pointer to the `IndexingField` object.

**Return Values**

This method has no specific return values. It returns standard return values of ABBYY Recognition Server COM-based API functions.

**See also**

Working with Collections.
ABBYY Recognition Server 4 COM-based API
BooleanIndexingField Object (IBooleanIndexingField Interface)

This object represents an indexing field of boolean type. It inherits from IIndexingField interface and has the same properties.

See also

IndexingField, IndexingFields.
ABBYY Recognition Server 4 COM-based API
EnumerationIndexingField Object
(IEnumerationIndexingField Interface)

This object represents an indexing field of enumeration type. It inherits from \texttt{IIndexingField} interface and has the same properties.

Additional Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PossibleValues</td>
<td>\texttt{StringsCollection}, read-only</td>
<td>Returns list of possible values of the indexing field.</td>
</tr>
</tbody>
</table>

See also

\texttt{IndexingField}, \texttt{IndexingFields}. 
ABBYY Recognition Server 4 COM-based API
MultipleLineIndexingField Object (IMultipleLineIndexingField Interface)

This object represents an indexing field of multiple lines. It inherits from `IIndexingField` interface and has the same properties.

See also

[IndexingField](#),
[IndexingFields](#).
ABBYY Recognition Server 4 COM-based API
RegularExpressionIndexingField Object
(IRegularExpressionIndexingField Interface)

This object represents an indexing field of Regular Expression type. It inherits from `IIndexingField` interface and has the same properties.

Additional Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expression</td>
<td>String</td>
<td>Regular expression for the indexing field.</td>
</tr>
</tbody>
</table>

See also

- `IIndexingField`
- `IndexingFields`
SingleLineIndexingField Object
(ISingleLineIndexingField Interface)

This object represents an indexing field of single line. It inherits from IndexingField interface and has the same properties.

See also

IndexingField,
IndexingFields.
ABBY Recognition Server 4 Web Services API
Connect Method of Client Object

This method establishes a connection with the server.

**Visual Basic Syntax**

Method `Connect(
    name As String
)

**C++ Syntax**

`HRESULT Connect(
    BSTR name
);

**Parameters**

`name`

[in] This parameter contains the DNS name of the server or its IP address.

**Return Values**

This method has no specific return values. It returns standard return values of ABBYY Recognition Server Open API functions.

**See also**

Client.

**See samples:** Hello, Listening, AsyncProcessing, Sample for ASP.NET.
ABBY Recognition Server 4 COM-based API
ProcessFile Method of Client Object

This method uses the parameters of the specified workflow to recognize the specified input image.

Visual Basic Syntax

Method ProcessFile(
    fileName As String,
    workflowName As String,
    password As String
) As XmlResult

C++ Syntax

HRESULT ProcessFile(
    BSTR fileName,
    BSTR workflowName,
    BSTR password,
    IXmlResult** result
);

Parameters

fileName
[in] This parameter contains the image file name.

workflowName
[in] This parameter contains the workflow name. The workflow name must be an element of the list returned by the IClient::Workflows property.

password
[in] This parameter contains the password for accessing the PDF file. If a non-PDF file is being recognized, this parameter is ignored. The default value is empty string.

result
[out] A pointer to an IXmlResult* pointer variable that receives the interface pointer of the XmlResult object.

Return Values

This method has no specific return values. It returns standard return values of ABBYY Recognition Server Open API functions.

See also

IClient,
IClient::ProcessFileAsync.
See sample: Hello.
ABBYY Recognition Server 4 COM-based API
IsListening Method of Client Object

This method allows you to learn whether receipt of job completion notifications from a selected workflow is enabled or disabled. The StartListening starts or stops the receipt of notifications. Between the "start receipt" call and the "stop receipt" call of the StartListening method, the IsListening method returns TRUE.

Visual Basic Syntax

Method IsListening(
    workflow As String
) Boolean

C++ Syntax

HRESULT IsListening(
    BSTR workflow
    VARIANT_BOOL* result
);

Parameters

workflow
    [in] This parameter contains the workflow name. The workflow name must be an element of the list returned by the IClient::Workflows property.

result
    [out] A pointer to a Boolean variable that specifies whether receipt of job completion notifications from a selected workflow is enabled or disabled. Must not be NULL.

Return Values

This method has no specific return values. It returns standard return values of ABBYY Recognition Server Open API functions.

See also

Client,
IClient::StartListening.

See sample: Listening.
ABBYY Recognition Server 4 COM-based API
StartListening Method of Client Object

This method allows you to get notifications about the completion of work by a specified workflow. The `doStart` parameter starts or stops the receipt of notifications. If the method is repeatedly called for the same workflow with the same value of the `doStart` parameter (twice "start receipt" or twice "stop receipt"), redundant calls are ignored. You will receive notifications about jobs which are completed between the "start receipt" call and the "stop receipt" call and whose resulting XML files are saved into the Output folder. If you want the resulting XML file to be saved into the Output folder, you must select the **Save XML Result file in folder** option on the **Output** tab of the **Workflow Properties** dialog box.

The **IsListening** method allows you to learn whether receipt of notifications from a selected workflow is enabled or disabled.

**Visual Basic Syntax**

Method `StartListening(
    workflow As String,
    doStart As Boolean
)

**C++ Syntax**

```cpp
HRESULT StartListening(
    BSTR workflow,
    VARIANT_BOOL doStart
);
```

**Parameters**

`workflow`
- [in] This parameter contains the workflow name. The workflow name must be an element of the list returned by the `IClient::Workflows` property.

`doStart`
- [in] This variable of the Boolean type starts or stops the receipt of notifications.

**Return Values**

This method has no specific return values. It returns **standard return values of ABBYY Recognition Server Open API functions**.

**See also**

[Client](#)
IClient::IsListening.

See sample: Listening.
OnJobComplete Method of IClientEvents Interface

This method is implemented on the client side. It is called by the ABBYY Recognition Server COM-based API when some of the methods of the Client object are running. It provides information about completion of the job with the specified ID to the client.

Job ID can be changed during job processing, e.g. if document separation is turned on in the workflow settings. In this case, to receive notifications about job completion, you should first start the receipt of notifications from the corresponding workflow with the help of the StartListening method of the Client object. This workflow must publish XML Result files for processed jobs to the Output folder.

**Visual Basic Syntax**

Sub OnJobComplete(  
    ByVal jobId As String,  
    ByVal result As XmlResult  
)

**C++ Syntax**

HRESULT OnJobComplete(  
    BSTR jobId,  
    IXmlResult* result  
);

**Parameters**

**jobId**

[in] This variable contains the job ID.

**result**

[in] This variable refers to the XmlResult object that contains the parameters and results of processing one job.

**Remarks**

The client implementation of this method must assure that all exceptions thrown inside the method are caught and handled and no exceptions are propagated outside the method. Propagation of an exception outside the method may lead to unpredictable results (such as program termination).

**See also**
IClientEvents, Client.

See samples: Listening, AsyncProcessing.
ABBYY Recognition Server 4 Web Services API
ProcessFileAsync Method of Client Object

This method uses the parameters of the specified workflow to recognize the input image in asynchronous mode.

Visual Basic Syntax

Method ProcessFileAsync(
    fileName As String,
    workflowName As String,
    password As String
) As String

C++ Syntax

HRESULT ProcessFileAsync(
    BSTR fileName,
    BSTR workflowName,
    BSTR password,
    BSTR* jobId
);

Parameters

fileName
    [in] This parameter contains the image file name.

workflowName
    [in] This parameter contains the workflow name. The workflow name must be an element of the list returned by the IClient::Workflows property.

password
    [in] This parameter contains the password for accessing the PDF file. If a non-PDF file is being recognized, this parameter is ignored. The default value is an empty string.

jobId
    [out] A pointer to a string variable that receives the job identification.

Return Values

This method has no specific return values. It returns standard return values of ABBYY Recognition Server Open API functions.

See also

Client, IClient::ProcessFile.
See sample: AsyncProcessing.
ABBYY Recognition Server 4 Web Services API
CreateXmlTicket Method of Client Object

This method creates an `XmlTicket` object based on the specified workflow.

Visual Basic Syntax

Method CreateXmlTicket(
  workflowName As String
) As XmlTicket

C++ Syntax

HRESULT CreateXmlTicket(
  BSTR workflowName
  IXmlTicket** result
);

Parameters

`workflowName` [in] This parameter contains the workflow name. The workflow name must be an element of the list returned by the `IClient::Workflows` property.

`result` [out] A pointer to an `IXmlTicket*` pointer variable that receives the interface pointer of the `XmlTicket` object.

Return Values

This method has no specific return values. It returns standard return values of ABBYY Recognition Server Open API functions.

See also

`Client`, `XmlTicket`.

See sample: Sample for ASP.NET.
**ProcessXmlTicket Method of Client Object**

This method uses the parameters of Xml Ticket and the specified workflow to recognize the input image. The Xml Ticket parameters override the workflow parameters.

**Visual Basic Syntax**

```vbnet
Method ProcessXmlTicket(
    ticket As XmlTicket,
    workflowName As String
) As XmlResult
```

**C++ Syntax**

```cpp
HRESULT ProcessXmlTicket(
    XmlTicket* ticket,
    BSTR workflowName,
    IXmlResult** result
);
```

**Parameters**

- `ticket`
  
  [in] This variable refers to the `XmlTicket` object that corresponds to the processing parameters of one job.

- `workflowName`
  
  [in] This parameter contains the workflow name. The workflow name must be an element of the list returned by the `IClient::Workflows` property.

- `result`
  
  [out] A pointer to an `IXmlResult`* pointer variable that receives the interface pointer of the `XmlResult` object.

**Return Values**

This method has no specific return values. It returns standard return values of ABBYY Recognition Server Open API functions.

**See also**

- `Client`
- `IClient::ProcessXmlTicketAsync`

**See sample:** Sample for ASP.NET.
AddImage Method of XmlTicket Object

This method adds an image to XmlTicket.

Visual Basic Syntax

Method AddImage(
    fileName As String
    password As String
)

C++ Syntax

HRESULT AddImage(
    BSTR fileName
    BSTR password
);

Parameters

fileName
    [in] This parameter contains the file name.
password
    [in] This parameter contains the password for accessing the PDF file. The default value is an empty string.

Return Values

This method has no specific return values. It returns standard return values of ABBYY Recognition Server Open API functions.

See also

XmlTicket.

See sample: Sample for ASP.NET.
ABBYY Recognition Server 4 Web Services API
Page Object

This object represents a single page of the input image file.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>String, read-only</td>
<td>Stores the unique identifier of the page. The identifier is generated automatically.</td>
</tr>
<tr>
<td>Number</td>
<td>String, read-only</td>
<td>Stores the page index in the input image file. The indexing starts with 0.</td>
</tr>
</tbody>
</table>

See also

Pages.
ABBYY Recognition Server 4 Web Services API
### Standard Return Codes of ABBYY Recognition Server COM-based API Functions

Below is a list of the standard return codes of the ABBYY Recognition Server Web Services API functions and properties.

<table>
<thead>
<tr>
<th>Return code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>S_OK</td>
<td>Method completed successfully.</td>
</tr>
<tr>
<td>E_OUTOFMEMORY</td>
<td>There was not enough memory to perform the operation.</td>
</tr>
<tr>
<td>E_UNEXPECTED</td>
<td>Unexpected internal error.</td>
</tr>
<tr>
<td>E_NOTIMPL</td>
<td>Method is not implemented.</td>
</tr>
<tr>
<td>E_POINTER</td>
<td>Invalid pointer argument.</td>
</tr>
<tr>
<td>E_INVALIDARG</td>
<td>One or more arguments are invalid.</td>
</tr>
<tr>
<td>CO_E_OBJNOTCONNECTED</td>
<td>A pointer to an object was passed that is no longer valid (this object was destroyed).</td>
</tr>
<tr>
<td>E_FAIL</td>
<td>Unspecified error.</td>
</tr>
</tbody>
</table>

Other return codes are possible, specifically those related to file system errors.
PagePosition Object

This object stores information about the page position within the input document.

Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FileId</td>
<td>String, read-only</td>
<td>Stores the unique identifier of the input file. The identifier is generated automatically. The value of this property is the same as the value of InputFile.ID.</td>
</tr>
<tr>
<td>PageId</td>
<td>String, read-only</td>
<td>Stores the unique identifier of the page. The identifier is generated automatically. The value of this property is the same as the value of Page.ID.</td>
</tr>
</tbody>
</table>

See also

PagePositions.
The **Client** object in the ABBYY Recognition Server Open API is a so-called "connectable object". This means that it implements the **IConnectionPointContainer** interface. A connectable object ensures communication between the Open API and its clients.

This connectable object provides connection points of two types — one that uses a **dispatch** interface, and one that uses an interface derived from **IUnknown**. The **dispatch** interface is designed for automatic use in Visual Basic and similar environments, while the vtbl-based interface is suitable for use in C++.

An Open API client application that needs to receive notifications of certain events in the Open API must implement interfaces of a specific type and "advise" objects implementing these interfaces to the corresponding connectable objects.

In Visual Basic this is done by simply declaring the object ** WithEvents** and implementing the corresponding methods of the callback interface. The procedure for Visual Basic is described in the **IClientEvents** interface.

Below follows an example of connecting an object on the client side.

```cpp
class CClientEventsListener : public IClientEvents {
public:
    ...

    // Provide simple implementation of IUnknown methods. They may also be implemented through inheritance from some standard class with COM support
    ULONG AddRef();
    ULONG Release();
    HRESULT QueryInterface(REFIID iid, void** ppvObject)
    {
        if (ppvObject == 0)
            return E_INVALIDARG;

        if (riid == __uuidof(IClientEvents)) {
            *ppvObject = static_cast<IClientEvents*>(this);
        } else if (riid == IID_IUnknown) {
            *ppvObject = static_cast<IUnknown*>(this);
        } else {
            *ppvObject = 0;
            return E_NOINTERFACE;
        }

        AddRef();
        return S_OK;
    }

    // Provide IClientEvents method implementation
    HRESULT OnJobComplete(BSTR, IXmlResult*);
};
```
Thus we have a **CClientEventsListener** class that may be used to receive notifications from the **Client** object. The following section of code advises this object to the notifications source (error handling is omitted):

```cpp
// Suppose that we have already created the Client object
IClient* client;
IConnectionPointContainer* pContainer=0;
client->QueryInterface(IID_IConnectionPointContainer, (void**)&pContainer);
IConnectionPoint* pPoint=0;
pContainer->FindConnectionPoint(__uuidof(IClientEvents),
    &pPoint);
CClientEventsListener listener;
IUnknown* listenerUnknown=0;
listener.QueryInterface(IID_IUnknown, (void**)&listenerUnknown);
// A variable to store the cookie returned from the IConnectionPoint::Advise method
DWORD cookie;
pPoint->Advise(listenerUnknown, &cookie);
...
// After notification, the listener is no longer needed and should be unadvised
pPoint->Unadvise(cookie);
```

Refer to documentation on COM for a more detailed description of connectable objects.

**See also**

- **Client**
- **IClientEvents**
ABBYY Recognition Server 4 COM-based API
IClientEvents Interface

This is a callback interface that is used for reporting events from the Client object to the listeners. This interface is implemented on the client side. As it derives from the IUnknown interface, the client object should also implement the IUnknown methods. This interface is designed primarily for use in C++. Visual Basic users that wish to receive notifications from the Client object should declare it WithEvents and implement the following Subs:

Public WithEvents cl As ABBYYRecognitionServer.Client

Private Sub cl_OnJobComplete(ByVal jobId As String, ByVal result As XmlResult)
...
End Sub

**Note:** You cannot call the ProcessFile, ProcessFileAsync, ProcessXmlTicket, and ProcessXmlTicketAsync methods in the event handler.

### Methods

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OnJobComplete</td>
<td>Provides information about job completion to the client.</td>
</tr>
</tbody>
</table>

### Remarks

To receive notifications correctly, please make sure that your application releases control and runs the Windows message loop. See the sample below.

### Visual Basic 6.0

Dim WithEvents Client As ABBYYRecognitionServer.Client
Dim Workflow As String
Dim Filename As String

Private Sub StartListening_Click()
   ' The StartListening method is called and then control is released
   Client.StartListening Workflow, True
End Sub

Private Sub ProcessAsync_Click()
   ' The ProcessFileAsync method is called and then control is released
   Client.ProcessFileAsync Workflow, Filename
End Sub
Private Sub client_OnJobComplete(ByVal jobId As String, ByVal result As XmlResult)
    ' Here is a reaction to job completion
End Sub

See also

Client,
Working with Connectable Objects
ABBYY Recognition Server 4 Web Services API
**CreateDocumentAttribute Method of Client Object**

This method creates a `DocumentAttribute` object.

**Visual Basic Syntax**

```vbnet
def CreateDocumentAttribute(
    Type As AttributeTypeEnum
) As DocumentAttribute
```

**C++ Syntax**

```c++
HRESULT CreateDocumentAttribute(
    AttributeTypeEnum Type,
    IDocumentAttribute** Result
);
```

**Parameters**

*Type*
- [in] This variable specifies the type of the document attribute. See the `AttributeTypeEnum` description for the supported file formats.

*Result*
- [out] A pointer to an `IDocumentAttribute`* pointer variable that receives the interface pointer of the `DocumentAttribute` object.

**Return Values**

This method has no specific return values. It returns standard return values of ABBYY Recognition Server Open API functions.

**See also**

*Client*, *DocumentAttribute*
ABBYY Recognition Server 4 COM-based API
**CreateOutputFormatSettings Method of Client Object**

This method creates an **OutputFormatSettings** object.

**Visual Basic Syntax**

Method `CreateOutputFormatSettings`

```vbnet
Method CreateOutputFormatSettings(
    format As OutputFileFormatEnum
) OutputFormatSettings
```

**C++ Syntax**

```cpp
HRESULT CreateOutputFormatSettings(
    OutputFileFormatEnum format
    IOutputFormatSettings** result
);
```

**Parameters**

- `format`  
  [in] This variable specifies the format of the output file. See the **OutputFileFormatEnum** description for the supported file formats.

- `result`  
  [out] A pointer to an **IOutputFormatSettings** pointer variable that receives the interface pointer of the **OutputFormatSettings** object.

**Return Values**

This method has no specific return values. It returns standard return values of ABBYY Recognition Server Open API functions.

**See also**

- **Client**
- **OutputFormatSettings**
ABBYY Recognition Server 4 COM-based API
GetJobState Method of Client Object

This method allows you to get information about current state of the specified job in asynchronous mode.

Visual Basic Syntax

```visualbasic
Method GetJobState(
    JobId As String,
    ByRef State As JobStateEnum
    ByRef Progress As Long
)
```

C++ Syntax

```c++
HRESULT GetJobState(
    BSTR JobId,
    JobStateEnum* State
    Long* Progress
);
```

Parameters

**JobId**  
[in] This parameter contains the job ID. The job ID must be a string returned by the `IClient::ProcessFileAsync` or `IClient::ProcessXMLTicketAsync` method.

**State**  
[out] A pointer to a `JobStateEnum` variable that specifies current state of the job. Must not be NULL.

**Progress**  
[out] A pointer to a Long variable that specifies the job progress in a percent format. Must not be NULL.

Return Values

This method has no specific return values. It returns standard return values of ABBYY Recognition Server Open API functions.

See also

[Client](#),  
[JobStateEnum](#).
ABBYY Recognition Server 4 COM-based API
ProcessXmlTicketAsync Method of Client Object

This method uses the parameters of Xml Ticket and the specified workflow to recognize the input image in asynchronous mode. The Xml Ticket parameters override the workflow parameters.

Visual Basic Syntax

Method ProcessXmlTicketAsync(
    ticket As XmlTicket,
    workflowName As String
) As String

C++ Syntax

HRESULT ProcessXmlTicketAsync(
     XmlTicket* ticket,
     BSTR workflowName,
     BSTR* jobId
);

Parameters

ticket
    [in] This variable refers to the XmlTicket object that corresponds to the processing parameters of one job.
workflowName
    [in] This parameter contains the workflow name. The workflow name must be an element of the list returned by the IClient::Workflows property.
jobId
    [out] A pointer to a string variable that receives the job identification.

Return Values

This method has no specific return values. It returns standard return values of ABBYY Recognition Server Open API functions.

See also

Client, IClient::ProcessXmlTicket
ABBYY Recognition Server 4 COM-based API
UpdateWorkflow Method of Client Object

This method allows you to change settings of the specified workflow.

**Visual Basic Syntax**

```vbnet
Method GetJobState(  
    ByVal workflow As IWorkflow  
)
```

**C++ Syntax**

```cpp
HRESULT UpdateWorkflow(  
    IWorkflow* workflow  
);
```

**Parameters**

*workflow*

[in] A pointer to a Workflow variable that represents new settings of the workflow.

**Return Values**

This method has no specific return values. It returns standard return values of ABBYY Recognition Server Open API functions.

**See also**

Workflow.
ABBYY Recognition Server 4 COM-based API
ProcessXmlTicketAsync Method of Client Object

This method uses the parameters of Xml Ticket and the specified workflow to recognize the input image in asynchronous mode. The Xml Ticket parameters override the workflow parameters.

Visual Basic Syntax

Method ProcessXmlTicketAsync(
    ticket As XmlTicket,
    workflowName As String
) As String

C++ Syntax

HRESULT ProcessXmlTicketAsync(
    XmlTicket* ticket,
    BSTR workflowName,
    BSTR* jobId
);

Parameters

ticket
    [in] This variable refers to the XmlTicket object that corresponds to the processing parameters of one job.

workflowName
    [in] This parameter contains the workflow name. The workflow name must be an element of the list returned by the IClient::Workflows property.

jobId
    [out] A pointer to a string variable that receives the job identification.

Return Values

This method has no specific return values. It returns standard return values of ABBYY Recognition Server Open API functions.

See also

Client, IClient::ProcessXmlTicket
ABBYY Recognition Server 4 COM-based API
JobStateEnum

TextTypeEnum enumeration constants are used to describe the type of recognized text.

typedef enum {
    JS_NoSuchJob,
    JS_Waiting,
    JS_Paused,
    JS_Processing,
    JS_Verification,
    JS_VerificationWait,
    JS_Indexing,
    JS_IndexingWait,
    JS_Processed,
    JS_ProcessedPaused,
    JS_Publishing
} JobStateEnum;

Elements

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JS_NoSuchJob</td>
<td>There is no job with specified ID.</td>
</tr>
<tr>
<td>JS_Waiting</td>
<td>The job is waiting to be processed.</td>
</tr>
<tr>
<td>JS_Paused</td>
<td>The job is in the <strong>Paused</strong> state if corresponding workflow was stopped based on a schedule.</td>
</tr>
<tr>
<td>JS_Processing</td>
<td>The job is being processed.</td>
</tr>
<tr>
<td>JS_Verification</td>
<td>One or several pages of the job are being verified.</td>
</tr>
<tr>
<td>JS_VerificationWait</td>
<td>All pages of the job are waiting to be verified.</td>
</tr>
<tr>
<td>JS_Indexing</td>
<td>Document of the job is being indexed.</td>
</tr>
<tr>
<td>JS_IndexingWait</td>
<td>Document of the job is waiting to be indexed.</td>
</tr>
<tr>
<td>JS_Processed</td>
<td>The job has been processed, but has not yet been published.</td>
</tr>
<tr>
<td>JS_ProcessedPaused</td>
<td>The job cannot be published.</td>
</tr>
<tr>
<td>JS_Publishing</td>
<td>The job is being published.</td>
</tr>
</tbody>
</table>

See also

IClient::GetJobState