



Open Practice Protocol

DTX Studio integration for PMS

Revision	Protocol	Author	Description of change
0	v1.0	Jasper Leemans	Original version.
1	v1.1	Jasper Leemans	Changelog: <ul style="list-style-type: none"> - "UPDATE" command now allowed in combination with -p, -P via command-line interface. - <data> elements now allowed for the "UPDATE" command via the file-based interface.
2	v1.2	Jasper Leemans, Nathan Steurs	Changelog: <ul style="list-style-type: none"> - Added list of known OPP implementers. - Added a paragraph about versioning. - Added ACQUIRE command. - Added UPDATE_CREATE command. - Added VIEW_DATA command. - Added InstallDir registry key to get the install location of the .exe's.
3	v1.3	Jasper Leemans	Changelog: <ul style="list-style-type: none"> - Added the forceOpen flag to the XML based implementation. - Added the VIEW command to the command-line and XML based implementations.
4	v1.3 rev. 2	Jasper Leemans	Changelog: <ul style="list-style-type: none"> - Added the -d/-D command line option to pass in the date format for PMS systems that don't support the ISO date format.
5	v1.3 rev. 4	Jasper Leemans	Changelog: <ul style="list-style-type: none"> - Removed forceOpen flag since it was never used and not really needed - Added options to pass a patient middle name

Introduction	7
Versioning	8
Configuration	9
Sending commands (Invoker)	10
<hr/>	
Command-line interface	10
OPP Command (-c, -C)	12
Silent flag (-s, -S)	12
Patient Reference ID (-r, -R).....	13
Practice patient ID (-i, -I)	13
Path to file or folder (-p, P).....	13
Date Format (-d, D).....	13
Path to file or folder (-p, -P).....	13
File-based interface	13
Exit codes	14
General/unknown error (1)	15
Misuse of shell interface and/or interface not implemented (2).....	15
Reference not found [Reference ID] (3)	15
Could not write (4)	15
Command not supported (5)	16
Reference already exists (6).....	16
Invalid data (7).....	16
Reserved for later [8, 125].....	16
Bash special meaning [126,255]	16
Receiving commands (Receiver)	17
<hr/>	
Command-line trigger	17
File-based trigger	17
Exit codes	17
XML file schema	19
<hr/>	
Version, encoding	19
Body <opp>	19
Command <command>.....	19
forceOpen	20
File-based commands	20

Software application <software>	21
Patient information <patient>	22
Patient notes <note>	23
Patient (medical) data <data>	23
Data reference ID	24
Data types	24
Supported image formats	26
Device <device>	26
Level	26
Window	26
Hard Tissue threshold	27
Scanned region <scannedRegion>	27
Tooth <tooth>	27
Anatomy <anatomy>	27
Known implementers	29
<hr/>	
DTX Studio	30
Enable PMS integration	30
Windows	30
Mac	30
Credits	31
References	32
<hr/>	

Introduction

This document specifies the Open Practice Protocol as implemented by the software (DTXSync). This is an open PMS interface owned and maintained by us.

The **Open Practice Protocol** (OPP) aims to be an **open** communication protocol between **practice management software (PMS)** and patient **diagnostic / treatment software (DTS)**. The protocol is – open – in the sense that its specifications are freely available and can thus be implemented by multiple vendors.

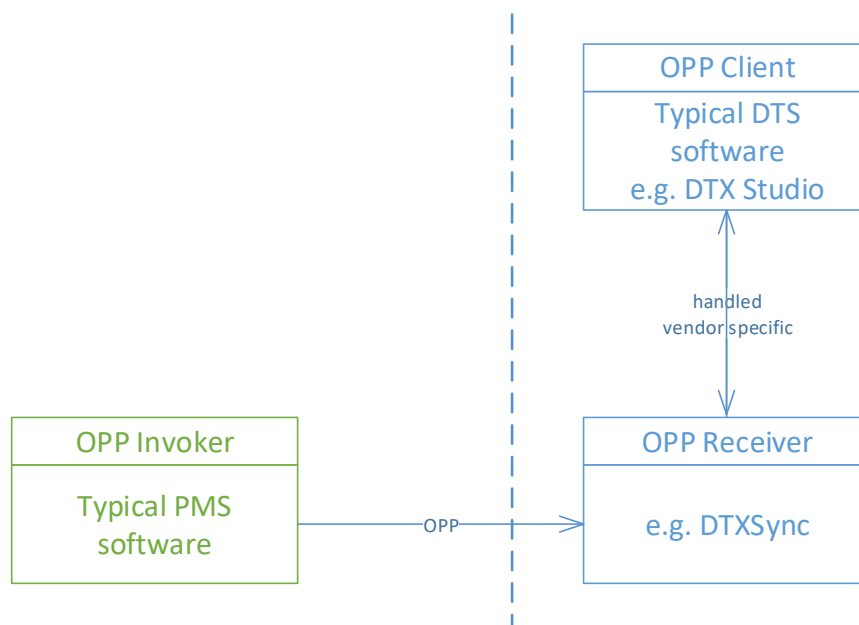
The scope of this document is to describe the Open Practice Protocol. The document provides the required input for being able to implement the protocol as a software maker. The OPP defines both a simplified command-line interface as well as a more extended file-based approach.

Versioning

Different versions of the Open Practice Protocol specification will exist at some point. And different applications will have a different (minimum) versions implemented. For this reason, we keep new versions backwards compatible with previous versions, in that case the second versioning number will go up $x.y$. If we would introduce an API breaking change at some point, the first number would go up $x.y$. We also encourage implementers to be forgiving: when an element or attribute is unknown to your implementation simply ignore it, don't fail on execution.

Configuration

The Open Practice Protocol has a fixed setup of three software applications. We identify them as the OPP **Client**, the OPP **Receiver** and the OPP **Invoker**. Depending on the kind of software your company is making, you might want to implement only the Receiver/Client end of the protocol or only the Invoker part (or both). The communication between the OPP Receiver and the OPP Client is **not** a part of the protocol definition as outlined in the drawing below. Typically, the OPP Receiver is installed together with the OPP Client.



Sending commands (Invoker)

This chapter covers what you, as a software vendor, need to do to send OPP commands to a Receiver with your **Invoker** application.

First of all, the Invoker application must be made configurable to point to the correct Receiver executable for each user action that should trigger an OPP command.

For example, for sending commands to DTX Studio (an OPP Client) the user of the OPP Invoker application must be able to point to the installed DTX Studio Receiver application.

```
C:\Program Files\DTX Studio Clinic\DTXsync.exe
```

This is an example location; the Receiver executable can be anywhere on the system as long as it's accessible by the user running the Invoker software.

Next, the implementer of the Invoker application should run the Receiver application with the correct parameters for the corresponding user action. For a basic collaboration between applications the **command-line interface** can be used. For a more advanced interoperability the **file-based approach** is recommended.

It is the responsibility of the Invoker application to correctly catch the **exit code** of the Receiver and to correctly communicate to the user any mishaps that might occur.

COMMAND-LINE INTERFACE

The command-line interface that is part of the OPP enables a minimalistic level of collaboration between the software applications. It's format and capabilities are the following.

```
> DTXSync.exe -c/C {string} -s/S -r/R {string} -f/F {string} -l/L {string}
-b/B {date} -d/D {date format} -g/G {string} -i/I {string} -p/P {string}
```

Below, all the command line parameters that a Receiver should support are documented in detail. Also note that the parameters can be either **mandatory or optional** based on the OPP **command** you are trying to invoke.

Letter	Parameter	Mandatory	Optional	Example
-c, -C	OPP command	Always	/	-c "CREATE"
-s, -S	Silent flag	/	CREATE, UPDATE, DELETE	-s
-r, -R	Patient Reference ID	Always	/	-r "c19ea492ca35"
-f, -F	First name	CREATE	UPDATE	-f "Rick"
-m, -M	Middle name	/	CREATE, UPDATE	-m "George"
-l, -L	Last name	CREATE	UPDATE	-l "Pickle"
-b, -B	Birthdate: [ISO 8601: YYYY-MM-DD]	CREATE	UPDATE	-b "1985-07-25"
-d, -D	Date format: yyyy for year, MM for month, dd for day including separators	/	/	-d "dd/MM/yyyy"
-g, -G	Gender: [MALE, FEMALE, OTHER]	CREATE	UPDATE	-g "MALE"
-i, -I	Practice patient ID	/	CREATE, UPDATE	-i "JLEE1985"
-p, -P	Absolute path to file or folder	/	CREATE, UPDATE	-p "path/to/dicom/folder/"

OPP Command (-c, -C)

This obligatory field specifies which OPP action the Receiver/Client combo needs to fulfil. The following commands are possible via the command-line interface:

Command	Explanation
CREATE	Creates the patient in the OPP Client application.
UPDATE	Updates an existing patient in the OPP Client application.
UPDATE_CREATE	Tries to update an existing patient in the OPP Client application, if the patient is not found it will automatically create the patient. Take note that all information needed to create the patient must be passed.
OPEN	Opens/selects an existing patient in the OPP Client application.
DELETE	Deletes an existing patient in the OPP Client application.
ACQUIRE	Opens the image acquisition module of the OPP Client application for the given patient.
VIEW	Opens an existing patient in the OPP Client (viewer) application.

Silent flag (-s, -S)

This optional flag, when turned on, specifies that the OPP Client application should NOT be started and/or brought to the foreground when the command is executed. It will obviously be ignored for the "OPEN" and "ACQUIRE" commands. Without the silent flag the Client application is supposed to present itself to the user (for all commands). E.g. CREATE/UPDATE commands will open the software with the newly created/updated patient opened or selected.

Patient Reference ID (-r, -R)

This obligatory field will be used by both the Invoker and the Receiver to **uniquely** identify a patient during any OPP communication. Typically, this is the patient's main identifier in your PMS database.

Practice patient ID (-i, -I)

This optional field can be used to store a **practice specific** patient identification scheme.

Path to file or folder (-p, P)

This optional field can be used to pass in a file path or folder. Note that an **absolute path** is expected here. The Receiver can either do something with it or ignore it, depending on the expected behaviour of the given OPP command. In case of the update commands the application will add the (DICOM) image or volume to the existing patient record.

Date Format (-d, D)

This optional field can be used to pass in a more specific date format according to the specifications defined in <https://doc.qt.io/qt-5/qdate.html#fromString-1>. If not specified, the default is the ISO 8601 date format: yyyy-MM-dd.

Path to file or folder (-p, -P)

This optional field can be used to pass in a file path or folder. Note that an absolute path is expected here. The Receiver can either do something with it or ignore it, depending on the expected behaviour of the given OPP command. In case of the update commands the application will add the (DICOM) image or volume to the existing patient record.

FILE-BASED INTERFACE

```
> DTXsync.exe {absolute/path/to/xml}
```

The file-based approach takes only a single command-line parameter, namely an XML file. This XML file has all the parameters the command-line interface supports but extends it with

more advanced options. Where the command-line interface is rather limited, the XML file can be easily extended in later versions of the OPP to add more features. The full schema for this XML file is fully documented in chapter 0 below.

IMPORTANT: The rule is that the Invoker application writes out the XML file to a location that is suitable for both the Invoker as well as the Receiver (in terms of access permissions). Typically, the system's TEMP directory is a good solution, but the path and file name can be freely chosen as long as the file extension ends in .XML and is passed as an absolute path. The Invoker is responsible for cleaning up the .XML after the Receiver returned its exit code.

EXIT CODES

When being run, the Receiver application must correctly output one of the following error codes when something went wrong. This way the Invoker can communicate any errors appropriately to the user.

	Receiver exit code
Success	0
General/unknown error	1
Misuse of shell interface and/or interface not implemented	2
Reference not found [Reference ID]	3
Could not write	4
Command not supported	5
Reference already exists [Reference ID]	6
Invalid data	7
Reserved for later	[8, 125]

Bash special meaning	[126,255]
----------------------	-----------

General/unknown error (1)

The OPP command was not successfully executed by the Receiver application. For example, this could be because the XML file could not be parsed or an invalid value was passed to the command-line arguments (e.g. non-existing Command).

Misuse of shell interface and/or interface not implemented (2)

The implementer of the Receiver application is allowed to implement only the **command-line interface** or the **file-based interface** of the protocol. When implementing only one of the two, the application should return this error code when somebody tries to invoke it in the other way. Of course, we recommend implementing both interfaces for more flexibility.

Other than that, invalid command-line parameters (e.g. non-existing letters), also yield this error.

Reference not found [Reference ID] (3)

For the commands [UPDATE, OPEN, DELETE, ACQUIRE, VIEW] it can very well be that the patient was never created, or doesn't exist anymore, on the Client side. In that case this error code is returned. The Invoker can also use this code to check if he still needs to [create] the patient or not. In the case of the [VIEW_DATA] command it can also mean that an unknown data reference was passed.

Could not write (4)

It could be that the Receiver application cannot make modifications to the patient [UPDATE, DELETE] due to the fact that the Client has that specific patient currently open. Since no modifications could be done this should definitely be relayed to the Invoker application via a special return code.

Command not supported (5)

This error code is returned when trying to execute a command that is not supported by the Receiver/Client. This could be the command in its entirety (e.g. UPDATE) or a specific extension to it (e.g. UPDATE with new medical images).

Reference already exists (6)

This error code is returned when trying to execute the CREATE command for a patient that is already known. Or on UPDATE when adding a data item that already exists under the given patient (the latter for file-based implementations only!).

Invalid data (7)

This error code is returned when one or more data items could not be loaded. This can be due to an unsupported file format or an invalid file path. In case of a CREATE/UPDATE it can be that other (valid) changes were correctly made to the patient record.

Reserved for later [8, 125]

These return codes are still free and can be used later if we need more specific feedback.

Bash special meaning [126,255]

Our return codes are based on the commonly known Bash Exit codes [\[10\]](#). Here it is specified that this range has a special meaning and should not be used for anything else. Even though we don't implement them, we stick to the Bash specifications.

Receiving commands (Receiver)

This short chapter covers how you, as a software vendor, need to implement an OPP compliant **Receiver** application for your specific **Client** application.

An OPP compliant Receiver application can have any name, however it is recommended to keep it static across releases for compatibility. For example, NobelClinician/DTX Studio Implant (Client) has an OPP compliant receiver called NCSync.exe. We advise to always install the Receiver together with the Client application. Hence, on Windows systems the Receiver .exe will typically be installed in the same **Program Files** folder as the Client application. On OS X (Mac) systems the Receiver executable can be embedded in the **Application Bundle** (.app) of your installed Client software application.

IMPORTANT: The Receiver application is by no means your actual Client application but acts as a proxy between your Client and the Invoker issuing the OPP commands. Hence, it is expected to be small, lightweight and very responsive.

COMMAND-LINE TRIGGER

Just implement the specifications as outlined in paragraph 0. When your process is finished, which should be in a timely manner, throw the correct exit code.

File-based trigger

For this interface, you only need to take one command-line parameter into account: the absolute file path to the XML, see paragraph 0. However, you will need to implement an XML parser that is compatible with the specifications outlined in chapter 0. Remember that the Invoker writes the XML and is responsible for deleting it afterwards; when the Receiver has returned its exit code. This is true for both the "success" as well as any "fail" scenarios, see 0.

Exit codes

All exit codes that the Invoker expects the Receiver to throw are outlined in 0.

NOTE: It's good practice to have the Receiver application keep a local log file where more detailed error information can be logged. This is of course vendor specific and not defined in the standard; but a good idea none the less.

XML file schema

This paragraph outlines the XML schema for the file-based interface to OPP. When implementing the Invoker part of the OPP for the file-based interface one should make sure this schema is validated. For this we also provide an official external XML schema (.xsd file) and some example XML files. Lastly, the XML file should always be left in a human-readable state, so no encryption can be applied on it.

VERSION, ENCODING

The XML file is expected to use XML version 1.0 and UTF-8 encoding. This means the document should always start with the following line:

```
<?xml version="1.0" encoding="UTF-8"?>
```

BODY <OPP>

The body of the XML file starts with the **<opp>** element and can have the following attributes.

Attribute	Value	Type	Mandatory	Optional
version	The version of the OPP used.	string	Always	/

Example:

```
<opp version="1.0">  
...  
</opp>
```

COMMAND <COMMAND>

Inside the **<opp>** element the **<command>** element needs to be present. Here it's specified which OPP command should be executed by the Receiver. The **<command>** element can have the following attributes.

Attribute	Value	Type	Mandatory	Optional
type	OPP command	string	Always	/
silent	Silent flag (see 0): [YES, NO]	string	/	Always
forceOpen	Force open the software, even if a command could not be executed due to an error: [YES, NO]	string	/	Always

Example:

```
<command type="CREATE" silent="YES"/>
```

forceOpen

This option will still bring the software to the foreground and will try to open the item with the given Reference Id, even if something went wrong with processing the command. The correct Exit Code is still expected to be returned. For example, the software will still OPEN even though an unknown patient Reference Id was passed in. It is up to the Receiver application to handle that and communicate this issue to the user as it sees fit. This issue might contradict with the silent flag, but overrules it.

File-based commands

The following commands are available in the file-based implementation:

Command	Explanation
CREATE	Creates the patient in the OPP Client application.
UPDATE	Updates an existing patient in the OPP Client application.

UPDATE_CREATE	Tries to update an existing patient in the OPP Client application, if the patient is not found it will automatically create the patient. Take note that all information needed to create the patient must be passed.
OPEN	Opens an existing patient in the OPP Client application.
DELETE	Deletes an existing patient in the OPP Client application.
ACQUIRE	Opens the image acquisition module of the OPP Client application for the given patient.
VIEW	Opens an existing patient in the OPP Client (viewer) application.
VIEW_DATA	Opens a specific data item directly for viewing in the OPP Client (viewer) application.

SOFTWARE APPLICATION <SOFTWARE>

Inside the <opp> element the **<software>** element needs to be present. This element is filled in with the software information of the Invoker application. Hence the application that generated the XML file. The <software> element can have the following attributes.

Attribute	Value	Type	Mandatory	Optional
name	Name of the Invoker application.	string	Always	/
version	Version of the Invoker application.	string	Always	/

Example:

```
<software name="My first PMS software App" version="1.0.0.1"/>
```

PATIENT INFORMATION <PATIENT>

Inside the <opp> element the **<patient>** element needs to be present. This element is filled in with all the patient information or a subset thereof. The <patient> element can have the following attributes.

Attribute	Value	Type	Mandatory	Optional
reference	Reference ID (see 0).	string	Always	/
firstName	First name of the patient.	string	Create	Update
middleName	Middle name of the patient.	string	/	Create, Update
lastName	Last name of the patient.	string	Create	Update
birthdate	Birthdate: [ISO 8601: YYYY-MM-DD]	date	Create	Update
gender	Gender: [MALE, FEMALE, OTHER]	string	Create	Update
patientID	Practice patient ID (see 0).	string	/	Create, Update
clinicianFirstName	First name of the referral clinician.	string	/	Create, Update
clinicianLastName	Last name of the referral clinician.	string	/	Create, Update

Example:

```
<patient reference="PMS3489" firstName="Marty" middleName="George"
lastName="McFly" birthdate="1992-03-31" gender="MALE"
patientID="MCFLY1992" clinicianFirstName="John" clinicianLastName="Bunsen"
>
```

```
...  
</patient>
```

PATIENT NOTES <NOTE>

Inside the <patient> element the **<note>** element is **optional**. A note is like a “post-it” with some remark about the patient. Only one <note> element is allowed under the patient element. Currently a <note> doesn’t have any attributes. Hence, the content must be set via the text field of the element.

Example:

```
<note>Patient is a heavy smoker.</note>
```

PATIENT (MEDICAL) DATA <DATA>

For OPP Clients that might benefit from it, the protocol foresees that you can pass in other files as well. Typical examples are DICOM scans, clinical pictures, or other medical data. Links to these separate data files can be passed via the **optional <data>** element which resides under the <opp> element.

IMPORTANT: <data> elements are only useful in case of the create and/or update commands. They have no effect when running other commands. When you do pass them, in such case, they will be ignored by the Receiver end. In case of the update commands the application will add the data to the existing patient record.

A <data> element can have the following attributes.

Attribute	Value	Type	Mandatory	Optional
reference	Reference ID (see 0).	string	/	Always
type	Predefined set of types possible (see table below).	string	Always	/

path	Absolute file or folder path that points to the separate data files.	string	Always	/
direction	Direction specifier that specifies how the data was acquired: [FRONTAL, LATERAL]	string	/	Always
acquisitionDate	The timestamp at which the medical data was recorded. [ISO 8601: YYYY-MM-DDTHH:mm:ss]	date/time	/	Always

Example:

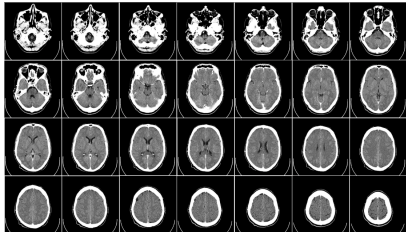
```
<data reference="IMG98656" type="CEPHALOGRAM" path="C:/example_ceph.dcm"
direction="LATERAL" acquisitionDate="1997-07-16T19:20:30">
...
</data>
```

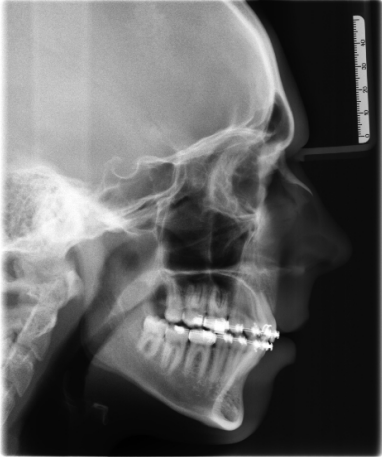



Data reference ID

The **reference ID** attribute for data is optional. The OPP Receiver/Client is allowed to internally generate an ID for the data item when not provided by the PMS. Starting from OPP v1.2, which has the VIEW_DATA command, the PMS has to provide a unique reference ID for the data item in order to identify it on both sides of the communication protocol.

Data types

The following data types are currently part of the specification.

Type	Description	Example	File	Folder
CT_DATA	Image set for a CT image volume. Typically one or more files in a DICOM format.		Yes	Yes

CEPHALOGRAM	<p>Single cephalometric radiograph.</p> <p>Typically single frame DICOM or image file.</p>		Yes	No
INTRAORAL	<p>Single intra-oral radiograph of a subset of teeth.</p> <p>Typically single frame DICOM or image file.</p>		Yes	No
PANORAMIC	<p>Panoramic radiograph, dental X-ray.</p> <p>Typically single frame DICOM or image file.</p>		Yes	No
PICTURE	<p>Generic clinical picture.</p> <p>Typically single color image file.</p>		Yes	No

Take into consideration that some data is coming in as a single file, while other data can contain a **set of files**. In the that case the Receiver should check whether it's a **file** or **folder** and act accordingly. For example, for data type **CT_DATA** (see table below) you can pass in either a single file (multi-frame) or a folder with a set of separate DICOM slices.

Supported image formats

For data types in the above table that handle single image files the most common image formats should be supported: DICOM, JPEG, PNG, BMP, TIFF.

DEVICE <DEVICE>

Inside the <data> element the **<device>** element is **optional**. This element can be used to pass information about the acquisition device that was used to acquire the data. It has a set of optional attributes that might help the OPP Client application to visualize the data in a better way.

Attribute	Value	Type	Mandatory	Optional
manufacturer	Manufacturer of the device.	string	Always	/
model	Device model name.	string	Always	/
level	Default image level threshold.	integer	/	Always
window	Default image window value.	integer	/	Always
hardTissueThreshold	Hard tissue (bone) threshold.	integer	/	Always

Example:

```
<device manufacturer="KAVO" model="OP3D" level="200" window="1100"
hardTissueThreshold="850"/>
```

Level

Only valid when the data is a greyscale image (single, volume) and "window" is also filled in.

Window.

Only valid when the data is a greyscale image (single, volume) and "level" is also filled in.

Hard Tissue threshold

The grayscale value in the image that matches the bone density. Only valid when the data is a grayscale image (single, volume). This value can for example be used by medical imaging software to segment bone out of the image.

SCANNED REGION <SCANNEDREGION>

Inside the <data> element the **<scannedRegion>** element is optional. This element specifies which region of the patient was acquired for that particular data item. The <scannedRegion> element currently has no attributes, only nested elements. An empty <scannedRegion> element is ignored by the Receiver.

Tooth <tooth>

Inside the <scannedRegion> element the **<tooth>** element is optional. This element specifies a tooth that is part of that particular data item. The text field of the <tooth> element contains a specific tooth number. Multiple <tooth> tags can be listed.

IMPORTANT: The numbers are in the **Universal Numbering System**.

Example:

```
<scannedRegion>  
  <tooth>4</tooth>  
  <tooth>5</tooth>  
  <tooth>6</tooth>  
</scannedRegion>
```

Anatomy <anatomy>

Inside the <scannedRegion> element the **<anatomy>** element is optional. This element specifies a specific anatomical structure of the patient that is part of that particular data item. The text field of the <anatomy> element contains one of the predefined anatomical structures as shown below. Multiple <anatomy> tags can be listed.

Anatomical structure	Description
CRANIUM	The skull base.
MAXILLA	The upper jaw.
MANDIBULA	The lower jaw.
CHIN	The chin.
LEFT_JAW_CONDYLE	The left condyle of the jaws.
RIGHT_JAW_CONDYLE	The right condyle of the jaws.
LEFT_MANDIBULAR_NERVE	The left nerve in the lower jaw.
RIGHT_MANDIBULAR_NERVE	The right nerve in the lower jaw.
LEFT_MANDIBULAR_RAMUS	The left ramus of the lower jaw.
RIGHT_MANDIBULAR_RAMUS	The right ramus of the lower jaw.

Example:

```
<scannedRegion>
  <anatomy>MANDIBULA</anatomy>
  <anatomy>LEFT_JAW_CONDYLE</anatomy>
  <anatomy>RIGHT_JAW_CONDYLE</anatomy>
</scannedRegion>
```

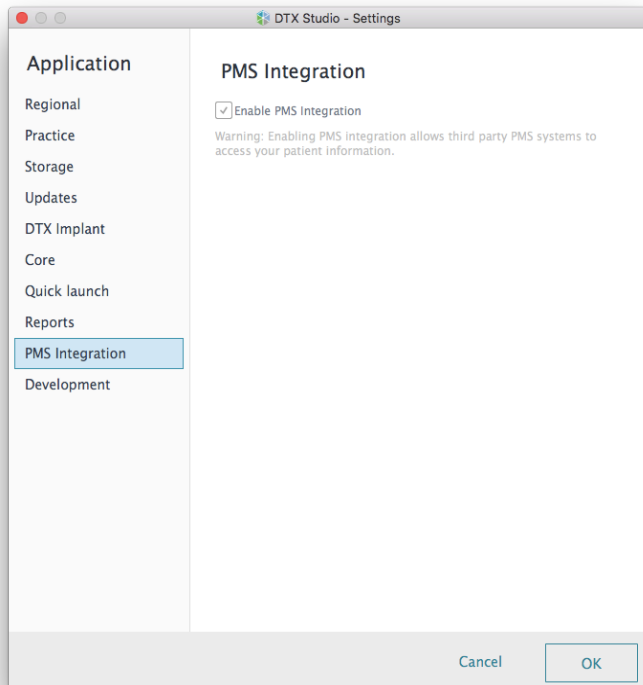
Known implementers

Below we list a table of known implementers of the OPP protocol. An OPP Invoker application can easily get the installed location of these by checking the supplied **registry keys**. Take note that no such equivalent keys exist on Mac, where we recommended looking into the default **/Applications** path.

OPP Client	OPP Receiver	Registry key install location
NobelClinician 3.2+	NCSync (OPP v1.1)	HKEY_LOCAL_MACHINE\SOFTWARE\NobelBiocare\NobelClinician\InstallDir
DTX Studio Implant 3.3+	NCSync (OPP v1.2)	HKEY_LOCAL_MACHINE\SOFTWARE\Medicim\DTXStudioImplant\InstallDir
DTX Studio Clinic 1.3+	DTXSync (OPP v1.1)	HKEY_LOCAL_MACHINE\SOFTWARE\Medicim\DTXStudioClinic\InstallDir
DTX Studio Clinic 1.5	DTXSync (OPP v1.2)	HKEY_LOCAL_MACHINE\SOFTWARE\Medicim\DTXStudioClinic\InstallDir
DTX Studio Clinic 1.6	DTXSync (OPP v1.3)	HKEY_LOCAL_MACHINE\SOFTWARE\Medicim\DTXStudioClinic\InstallDir
DTX Studio Clinic 1.6.10+	DTXSync (OPP v1.3 rev. 2)	HKEY_LOCAL_MACHINE\SOFTWARE\Medicim\DTXStudioClinic\InstallDir
DTX Studio Clinic 1.7.2+	DTXSync (OPP v1.3 rev. 4)	HKEY_LOCAL_MACHINE\SOFTWARE\Medicim\DTXStudioClinic\InstallDir

DTX STUDIO

Enable PMS integration



Windows

On Windows the OPP Receiver application (DTXSync.exe) is installed next to the Client application (DTXStudio.exe), by default this means you run it like:

```
C:\Program Files\DTX Studio Clinic\DTXSync.exe -c "CREATE" -r "PMSID" -f "Rick" -l "Pickle" -b "1985-07-25" -g "MALE"
```

Mac

On Mac the OPP Receiver application (DTXSync) is embedded in the Application Bundle of the DTX Studio application, which by default is installed in /Applications, so you run it like:

```
/Applications/DTXStudio.app/Contents/Applications/DTXSync.app/Contents/MacOS/DTXSync -c "CREATE" -r "PMSID" -f "Rick" -l "Pickle" -b "1985-07-25" -g "MALE"
```

Credits

The Open Practice Protocol is completely open for any software manufacturer to implement. This accounts for both the Invoker as well as the Receiver side. However, the **Medicim NV** company owns the current specification and any future updates made to it. If you have any remarks or request for a future version of the protocol, please contact Medicim NV directly.

References

[1] Medical practice management software,

https://en.wikipedia.org/wiki/Medical_practice_management_software

[2] CT scan,

https://en.wikipedia.org/wiki/CT_scan

[3] Cephalogram,

<https://en.wikipedia.org/wiki/Cephalogram>

[4] Dental anatomy,

https://en.wikipedia.org/wiki/Dental_anatomy

[5] Panoramic radiograph,

https://en.wikipedia.org/wiki/Panoramic_radiograph

[6] Universal Numbering System,

https://en.wikipedia.org/wiki/Universal_Numbering_System

[7] Ramus of the Mandible,

https://en.wikipedia.org/wiki/Ramus_of_the_mandible

[8] XML schema generator,

<http://www.freeformatter.com/xsd-generator.html>

[9] XML schema validator,

<http://www.xmlvalidation.com/>

[10] Bash Exit codes,

Bash Exit Codes With Special Meanings.pdf,

<http://tldp.org/LDP/abs/html/exitcodes.html>