

日医標準レセプトソフト クラウド版 push-exchanger 利用手順

ベンダー向け

2017 年 11 月 7 日

日本医師会 ORCA 管理機構株式会社

改版履歴

初版 2017 年 3 月 30 日

二版 2017 年 11 月 7 日 パス区切りを¥から/に修正

目次

1. 概要	1
2. push-exchanger 概要	1
3. push-exchanger の特徴	2
4. push-exchanger の動作概要	2
5. push-exchanger の設定	2
6. push-exchanger プラグイン仕様	3
7. 帳票印刷プラグイン print001	4
7-1. print001 プラグイン設定ファイル	5
7-2. 帳票作成プラグイン仕様	5
8. CLAIM プラグイン仕様	6
8-1. CLAIM プラグイン設定ファイル	7

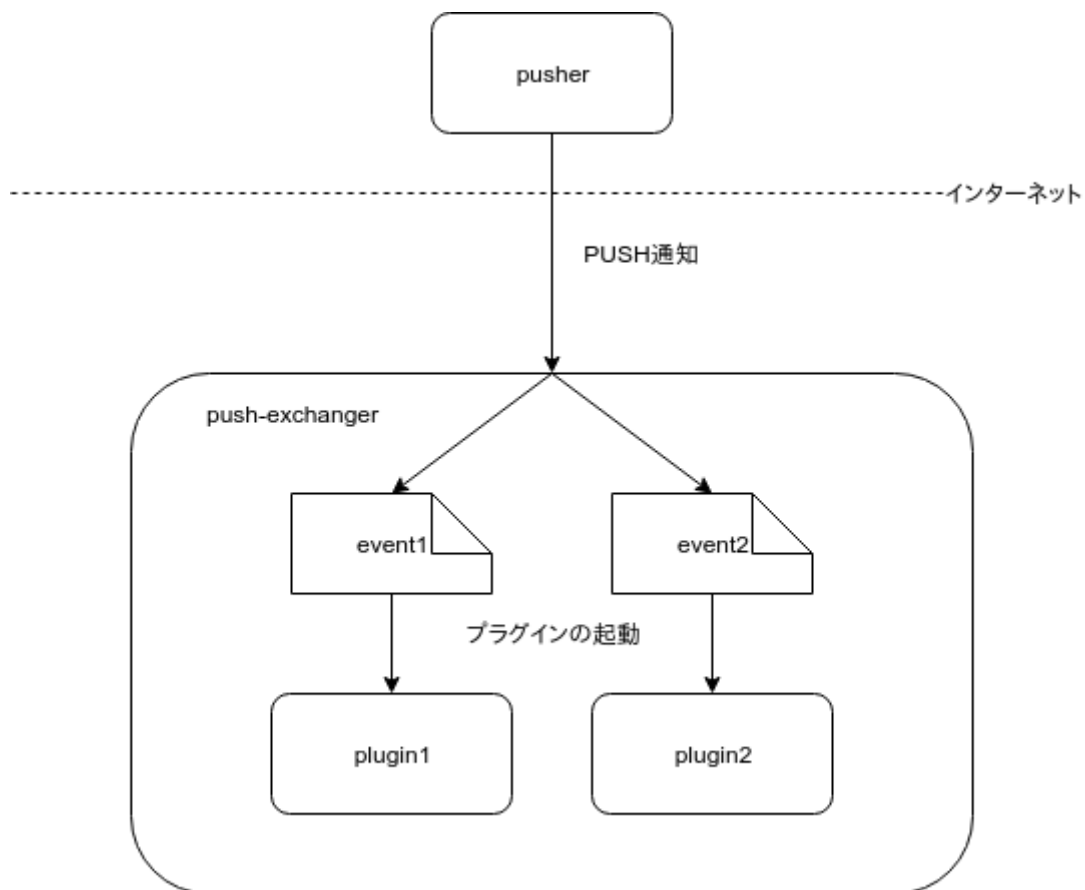
1. 概要

本文書では日レセ PUSH 通知駆動フレームワーク push-exchanger の仕様について記述する。日レセ PUSH 通知については日レセ PUSH 通知仕様書を参照。

2. push-exchanger 概要

push-exchanger は pusher(日レセ PUSH 通知サーバ)から日レセ PUSH 通知を受信し、PUSH 通知のイベントに対応したプラグインを起動するフレームワークである。

Push-exchanger のプラグインを利用したシステム連携および push-exchanger の仕組みを理解した push 通知の受信処理を開発するために役立てていただきたい。



3. push-exchanger の特徴

以下の特徴がある。

- ① Ruby で実装されている
- ② pusher との接続に WebSocket クライアントの faye/websocket というライブラリを利用している
- ③ pusher との接続が切断された場合に、10 秒間隔で再接続を試行する
- ④ Windows、Linux の環境で動作可能

4. push-exchanger の動作概要

push-exchanger の動作の概要は、以下のような流れになる。

- ① プラグインロードパスを参照しプラグインを読み込む
- ② pusher へ接続し subscribe を行う
- ③ PUSH 通知を受信する
- ④ PUSH 通知のイベントに対応するプラグインを起動する

5. push-exchanger の設定

push-exchanger の設定は YAML 形式で記述する。

項目名	説明	設定例
<code>:ws_server</code>	pusher の URI	<code>wss://pusher-proxy.orca.orcamo.jp/ws</code>
<code>:api_user</code>	日レセ API ユーザ	<code>ormaster</code>
<code>:api_key</code>	日レセ API キー	<code>xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx</code>
<code>:api_server</code>	日レセ API サーバ	<code>ap-proxy.orca.orcamo.jp</code>
<code>:api_port</code>	日レセ API サーバの ポート番号	<code>8080</code>
<code>:use_ssl</code>	SSL クライアント認証 の利用	<code>true</code>
<code>:ca_cert</code>	CA 証明書ファイル	<code>"c:/push-exchanger/certs/ca.crt"</code>
<code>:cert</code>	クライアント証明書	<code>"c:/push-exchanger/certs/tenant.crt"</code>
<code>:cert_key</code>	クライアント証明書 秘密鍵	<code>"c:/push-exchanger/certs/tenant.pem"</code>

:passphrase	秘密鍵パスフレーズ	xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
:log_file	ログファイル	"c:/push-exchanger/log/push-exchanger.log.txt"
:log_level	ログレベル	Info
:plugin_load_path	プラグインロードパス (配列)	- "c:/push-exchanger/plugin"

:ca_cert、:cert、:cert_key、:log_file、:plugin_load_path はダブルクォートで括った方がトラブルになりにくい。またパス区切り文字は「¥」ではなく「/」となっていることに注意が必要である。

6. push-exchanger プラグイン仕様

push-exchanger のプラグインの仕様は Ruby スクリプトで記述し、以下のように配置する必要がある。

:plugin_load_path/プラグイン名/プラグイン名.rb

例えば:plugin_load_path が"c:/push-exchanger/plugin"で、プラグイン名が test であったなら以下のように配置する。

c:¥push-exchanger¥plugin¥test¥test.rb

プラグインの Ruby スクリプトは以下の仕様を満たす必要がある。

- ① クラス名はプラグイン名と一致させる(プラグイン名が test とするならクラス名は Test)
- ② 対応するイベント名の配列を返す get_events を定義する
- ③ get_events で返したイベント名に対応するメソッドを定義し、その中でイベントに対応する処理を記述する

プラグインスクリプト例

```
require_relative "../pe-util"
require "pp"

# プラグイン名とクラス名の一致
class Test
```

```

def initialize(conf)
  @conf = conf
end

def get_events
  # acceptとaccountのイベントに対応
  %w|accept account|
end

# acceptのイベント処理
def accept(data)
  puts "test plugin accept"
  @conf[:logger].info("test accept")
end

# accountのイベント処理
def account(data)
  puts "test plugin account"
  @conf[:logger].info("test account")
end
end

```

7. 帳票印刷プラグイン print001

帳票印刷プラグインは、診療行為登録などの帳票発行のタイミングで発行される print001 イベントの PUSH 通知を受信して、API により帳票印刷データを取得し、それを元に帳票作成、印刷を行うプラグインである。大まかな動作は以下である。

- ① 日レセ PUSH サーバから print001 イベントを受信する
- ② PUSH 通知のユーザが有効ユーザでない場合、または無効ユーザであった場合は処理終了する
- ③ PUSH 通知の body 領域のレポート配列からレポート情報を取り出し逐次的に処理する
- ④ レポート情報から Data_ID、Form_ID、Custom_ID、Form_Name を取り出す
- ⑤ Data_ID をキーに /api01rv2/formdatagetv2 にアクセスし印刷情報を取得する
- ⑥ Custom_ID または Form_ID に対応した帳票作成プラグインを呼び出し PDF を作成する
- ⑦ 設定ファイルに指定されたプリンタから PDF を印刷する

7-1. print001 プラグイン設定ファイル

print001 プラグインの設定ファイルは YAML 形式で記述する。また print001.rb と同じディレクトリに print001.yml として配置する必要がある。以下、各設定項目について記述する。

項目名	説明	設定例
<code>:enable_users</code>	ユーザホワイトリスト	
<code>:disable_users</code>	ユーザブラックリスト	
<code>:printer_settings</code>	印刷設定(帳票名とプリンタ名のハッシュ) :default を指定した場合はデフォルトプリンタの設定	:default: "RICOH IPSiO NX85S RPDL" :karte_no1m: "CANON LBP8900"
<code>:report_dir</code>	帳票 PDF ファイル保存ディレクトリ	"c:/push-exchanger/tmp/print001"
<code>:report_limit</code>	帳票 PDF ファイル最大保存数	100
<code>:report_data_dir</code>	帳票データ保存ディレクトリ	"c:/push-exchanger/tmp/print001-data"
<code>:report_data_limit</code>	帳票データファイル最大保存数	100
<code>:plugin_load_path</code>	プラグインロードパス	"c:/push-exchanger/plugin/print001/plugin"
<code>:print_command</code>	Windows 環境印刷コマンド	"\"C:/Program Files (x86)/Adobe/Acrobat Reader DC/Reader/AcroRd32.exe\" /N /T \"%FILE%\" \"%PRINTER%\""

`:report_dir`、`:report_data_dir`、`:plugin_load_path`、`:print_command` はダブルクォートで括った方がトラブルになりにくい。またパス区切り文字は「¥」ではなく「/」となっていることに注意が必要である。

7-2. 帳票作成プラグイン仕様

プラグインは Ruby スクリプトである。`:plugin_load_path` に以下のように配置する

`:plugin_load_path/プラグイン名/プラグイン名.rb`

- ① プラグインの Ruby スクリプトは以下の仕様を満たす必要がある
- ② プラグイン名はカスタム ID(Custom_ID)または帳票 ID(Form_ID)と一致すること
- ③ クラス名はプラグイン名と一致させる(プラグイン名が abc とするならクラス名は Abc)
- ④ 印刷処理を行う export メソッドを定義すること
- ⑤ export メソッドはファイル名、帳票データ(JSON)、オプションの 3 つの引数を受け取る
 - オプションはハッシュで PushExchanger の設定(:conf)と print001.rb の設定(:pr_conf)が格納されている。
 - export メソッド内で帳票データを利用して引数のファイル名の PDF を作成する

プラグインスクリプトの例を以下に示す。

:plugin_load_path/okusuri_techo/okusuri_techo.rb

```
require "thinreports"
require "pp"

# プラグイン名とクラス名の一致
class Okusuri_techo

  # exportメソッドの定義
  def export(filename, report_data, option)
    # report_dataを利用してPDF(filename)を作成する
  end

end
```

8. CLAIM プラグイン仕様

CLAIM プラグインは CLAIM の PUSH 通知と CLAIM 情報取得 API を利用して CLAIM サーバのプロキシとなるプラグインである。 大まかな動作は以下である。

- ① 日レセ PUSH サーバから受付や診療行為登録などのタイミングで発行される CLAIM の PUSH 通知を受信する
- ② CLAIM PUSH 通知の内容を元に CLAIM 情報取得 API にアクセスして日レセ CLAIM 情報を取得する
- ③ CLAIM 情報を CLAIM XML テンプレートに埋め込み CLAIM XML を作成する
- ④ CLAIM XML を CLAIM 受信サーバ(電子カルテなどの連携機器)に送信する

8-1. CLAIM プラグイン設定ファイル

CLAIM プラグインの設定ファイルは YAML 形式で記述する。また `claim.rb` と同じディレクトリに `claim.yml` として配置する必要がある。次に、各設定項目について記述する。

項目名	説明	設定例
<code>:xml_log_dir</code>	送信 XML を保存するディレクトリ	<code>"c:/push-exchanger/tmp/claim_xml"</code>
<code>:xml_log_limit</code>	送信 XML ファイルを保存する最大数	100
<code>:servers</code>	CLAIM を送信するサーバの設定の配列、個々のサーバ設定はハッシュで記載する(後述)	

`:xml_log_dir` はダブルクォートで括った方がトラブルになりにくい。またパス区切り文字は「¥」ではなく「/」となっていることに注意が必要である。

`:servers` に記載するサーバ設定について説明する。

項目名	説明	設定例
<code>:name</code>	サーバ名	<code>server1</code>
<code>:host</code>	サーバの IP アドレス	<code>192.168.1.120</code>
<code>:port</code>	サーバのポート	<code>11111</code>
<code>:encoding</code>	文字エンコード	<code>UTF-8</code>
<code>:event</code>	送信するイベントの配列	<code>["accept","account"]</code>