HKIRC EPP API User Guide

Version 1.1.1 – 15 August 2017

This document is a concise guide to install and use the HKIRC EPP API. The HKIRC EPP API is a set of Java Libraries intended for any operating systems. The API creates Extensible Provisioning Protocol (EPP) commands that can be used to communicate with the HKIRC Registry EPP Gateway Server.

All EPP commands generated from the API need to be sent to the EPP gateway server using SSL.

Contents

Terms of Use

- HKIRC's EPP API –

Important Notice

These Terms of Use will apply when you use HKIRC's EPP API (defined in Clause 1 below). Before using HKIRC's EPP API, we ask that you read these Terms of Use carefully. If you find yourself unable to agree to these Terms of Use, then you must not use HKIRC's EPP API. By using HKIRC's EPP API, you accept and agree to be bound by these Terms of Use unconditionally.

15. In these Terms of Use, unless the context otherwise requires, the definitions of the following expressions are as follows:-

"Confidential Information"	means any information obtained from or through the Platform, including but not limited to, data relating to applicants and registrants of .hk domain names, whether in writing or otherwise that is not publicly known including any compilation of otherwise public information in a form not publicly known but not including information that, at the time of disclosure, is publicly known; information that, after disclosure, becomes publicly known other than as a result of a breach of this Terms of Use; information that the recipient can show was known to it prior to the disclosure; and information that the recipient can show was made known to it by a third party who was entitled to do so and who did not impose any obligation of confidentiality or restricted use;
"Domain Name Dispute Resolution Policy"	means HKIRC's Domain Name Dispute Resolution Policy;
"HKIRC"	means Hong Kong Internet Registration Corporation Limited;
"HKIRC's EPP API"	means HKIRC's Extensible Provisioning Protocol Application Programming Interface;
"Platform"	means HKIRC's API or SPAS II;
"Registration Agreement"	means HKIRC's Domain Name Registration Agreement for .hk Domain Names;
"Rules for .hk Domains and Sub-domains"	means HKIRC's Rules for .hk Domains and Sub-Domains; and
"SPAS II"	means Service Partner Administration System II.
"EPP"	means Extensible Provisioning Protocol
"IETF"	means Internet Engineering Task Force
"SDK"	means Software Development Kit
"SRS"	means Shared Registry System
"XML"	means Extensible Markup Language
"RGP"	means Registry Grace Period

2. By sending requests to us for Domain Name Administration through the Platform, you warrant HKIRC that you are aauthorizedaauthorized to apply for all the services

detailed in your requests on behalf of your customers, to legally bind your customers to the terms and conditions of the documents below and that you have notified your customers that their .hk domain name registrations will be bound by such documents, namely:
(i) the Registrar Agreement;
(ii) the Published Policies;
(iii) any other applicable agreements, rules, policies, Terms and Conditions and Important Notices.

3. You shall not explicitly or implicitly represent and/or cause any misconception to the public or to applicants or registrants of .hk domain names that you are the registrar or registry responsible for .hk domain names.

4. You shall not, whether by yourself, your agent, employees or any person aauthorizedaauthorized by you, do any of the following:

a. use any data obtained from the Platform to send out unsolicited commercial advertising of any sort via any medium including but not limited to, the telephone, email or fax; or
b. enable high volume, automated or electronic processes (except legitimate batch input processes for domain name registration and/or for maintenance purposes on the HKIRC's EPP API) to access HKIRC's computer systems including but not limited to, the Platform and WHOIS search enquiry service; or
c. compile, repackage, disseminate, disclose to any third party or use the data obtained from the Platform for any purpose other than obtaining information about a domain name registration record; or
d. use such data obtained from the Platform to derive an economic benefit for yourself or any third parties except by applying for or maintaining a .hk domain name for and on behalf of your customers.

5. You shall treat as confidential at all times all customer data and/or Confidential Information obtained from the Platform and such data shall not be used in any way detrimental to the customer and/or to HKIRC. You shall not divulge the customer data or Confidential Information to any person except to your own employees on a need-to-know basis. You shall prevent any authorized d use, dissemination or publication of Confidential Information and you shall not do any act, or engage in any practice, that contravenes the Hong Kong Personal Data (Privacy) Ordinance (Cap.486). In the event of any authorized d disclosure of protected personal data and/or Confidential Information by you, whether willfully or not, you and/or your representatives who disclose the data shall be wholly and personally liable for any losses and/or damages, whether monetary or not, which HKIRC may suffer directly and/or indirectly as a result of such authorized d disclosure. HKIRC shall be entitled to damages which include but not limited to, all legal costs and expenses incurred on HKIRC in connection with the enforcement of these Terms of Use.

6. HKIRC in its sole discretion may terminate or suspend your ability to access and/or use of the Platform at any time without prior notice including but not limited to, in the event of (i) a breach of these Terms of Use by you, (ii) non-performance of your obligations under these Terms of Use, (iii) breach by you of any of the Registration Agreement, HKIRC Service Partner Program Terms and Conditions.

7. DISCLAIMER. HKIRC ACCEPTS NO LIABILITY AND WILL NOT BE LIABLE FOR ANY LOSS OR DAMAGE ARISING DIRECTLY OR INDIRECTLY (INCLUDING SPECIAL, INCIDENTAL OR CONSEQUENTIAL LOSS OR DAMAGE) FROM YOUR ACCESS TO THE PLATFORM OR USE OF THE PLATFORM, NOT LIMITED TO BUT INCLUDING ANY LOSS, DAMAGE OR EXPENSE ARISING FROM ANY DEFECT, ERROR, IMPERFECTION OR INACCURACY WITH THE PLATFORM, ITS CONTENTS

OR ASSOCIATED SERVICES, OR DUE TO ANY UNAVAILABILITY OF ANY PART OF THE PLATFORM OR ASSOCIATED SERVICES OR TO ANY DELAY IN OPERATION OR TRANSMISSION, COMPUTER VIRUS, TROJAN HORSE, WORM, SOFTWARE BOMB, COMMUNICATION LINE FAILURE, SOFTWARE OR HARDWARE DATA (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR COMPATABILITY PROBLEMS), OR INTERCEPTION OF ON-LINE COMMUNICATION.

HKIRC MAKES NO REPRESENTATION OR WARRANTY OF ANY KIND, EXPLICIT, IMPLICIT OR STATUTORY, REGARDING THE PLATFORM OR THE MATERIALS AND INFORMATION CONTAINED OR REFERRED TO ON EACH PAGE ASSOCIATED WITH THE PLATFORM. NO CLAIMS OR ACTIONS OF LEGAL PROCEEDINGS IN CONNECTION WITH THE USE OF THIS PLATFORM BY YOU OR ANY THIRD PARTIES AUTHORISED BY YOU TO USE THE PLATFORM WILL BE ENTERTAINED BY HKIRC.

8. You, your employees, agents and any person authorizedaauthorized by you shall not seek to recover and shall not be entitled to recover from HKIRC or to be indemnified by HKIRC against any direct, indirect or consequential loss or damage suffered by you arising out of or in connection with your use of the Platform.

9. You shall be held responsible for any losses and damages suffered by HKIRC arising either directly or indirectly from the use of the Platform by you, your employees, agents or any person authorizedaauthorized by you. HKIRC reserves the right to claim, take any actions or legal proceedings for recovery or compensation as a result.

10. You hereby agree to fully indemnify, defend and hold HKIRC, its affiliates or partners, agents, directors, employees harmless from and against any action, liability, costs claim, loss, damage, proceeding or expense of whatsoever nature (including reasonable legal fees, costs and expenses on a full indemnity basis) arising from or directly or indirectly related to:-

(15) your access to and/or use of the Platform and/or any other person or entity's use of this service or the Platform where such person or entity was authorizedaauthorized by you to access and/or use the Platform; or

(b) any negligent act or omission or breach or non-observance of any of these Terms of Use by you or by any other person or entity where such person or entity authorizedaauthorized by you to use the Platform; or

© any materials you submit, post to, or transmit through the Platform.

11. All reasonable efforts are made by HKIRC to ensure the information provided on the Platform is accurate and/or up-to-date, HKIRC gives no warranty of any kind, explicitly or implicitly, with regard to the accuracy or completeness of any information on the Platform or that the Platform will be error free. Information obtained through the Platform may contain errors including but not limited to, inaccuracies or typographical errors. HKIRC shall not be liable to you for any loss arising out of and to the extent caused by any failure by HKIRC to keep the information on the Platform accurate and/or up-to-date. HKIRC reserves the right in its discretion but undertakes no duty to review, edit or otherwise change without prior warning or notice any information or material on the Platform.

12. HKIRC does not guarantee that any e-mails sent from the Platform will reach you or those you send to us will be received by HKIRC. HKIRC does not warrant the privacy and/or security of e-mails during Internet transmission.

13. HKIRC may u‟” ""session cook”es" on the Platform. If you want to disallow such cookies you can do so on your Web browser. These cookies are stored on your computer temporarily, not permanently, namely for the duration of each of your visits to the Platform.

14. HKIRC reserves the right to amend these Terms of Use at any time without prior notice. You shall review the SPAS II regularly for update.

15. These Terms of Use shall be governed by and construed in accordance with the laws of the Hong Kong Special Administrative Region.

16. In the event of any discrepancy between the English and Chinese versions of this Agreement, the English version shall prevail.

Company and Contact information

Hong Kong Internet Registration Corporation Limited
Unit 501, Level 5, Core C,
Cyberport 3, 100 Cyberport Road,
Hong Kong.

Email: registrar@hkirc.hk
Hotline No.: +852 2319 1313
Fax No.: +852 2319 2626
Office Hour: Monday to Friday, 9 a.m.- 6 p.m.

Change Log

Authors	Date	Description
SY ANG	5 Jul 2008	Initial Revision
SY ANG	5 Sep 2008	Updated content
SY ANG	8 Sep 2008	Added Parameter Column
Brent Lee	2 Aug 2010	Added bundling
Danny Chan	29 Jul 2011	Remove Auth Info field from Contact Object
Boon	15 Aug 2011	Update Query Tracking function
Danny Chan	10 Nov 2011	Added Maximum Length column for tables in Section 3.2.1.1 Create Contact and Section 3.2.1.2 Update Contact, to indicate the largest input size for each field in a contact.
Dan Chan	03 Feb 2012	Added tracking create RAC function
Dan Chan	14 Mar 2012	Added update tracking document status function and amended Transfer of Holding Rights function
Khoi Peng	14 Mar 2013	Added Add, Modify, and Remove Domain's Reseller Information function
Khoi Peng	17-Oct-2013	Amended NDN can add Domain's Reseller Information. Amended Domain Update XML can add exemption of Modify Domain (MDN) or Modify NS (MNS) or both for status update prohibited
Wing Wong	18-May-2016	Amended Domain Create to allow Promotion Code parameter passing
Andres Ip	03 July 2017	Add new method in Domain Update Object to support promotion for cross selling and brand protection.
Andres Ip	15 August 2017	1. Add new method in Domain Info Object to support DNSSEC (nothing updated in this document). 2. Add new method in Domain Update Object to support DNSSEC.

1 Introduction

1.1 Purpose

This document provides detailed guidelines on the usage of the Java SDK. Complete programming samples are provided to shorten integration and development time for registrars. Detail error codes and error messages are documented in the section below.

1.2 Contact and Name Server Policy Requirements

There are certain policies that are enforced in the HKIRC Registrar implementation of EPP:

A minimum of 4 contacts (including 1 Registrant/Owner, 1 of each Admin, Billing and Technical contacts) must be provided during the create domain transaction. All domains must be created with at least 2 name servers.

2 Prerequisites

Java SDK runtime 1.5 and above

The following packages are included in the HKIRC EPP API Startup Kit.

- eppsdk.jar
- junit-4.0.jar
- libidn-0.5.9.jar
- random.jar
- xercesImpl-2.6.0.jar
- xmlParserAPIs-2.6.0.jar

The HKIRC EPP API Startup Kit and all EPP related document are available to download from HKIRC Secure FTP server. Please contact registrar@hkirc.hk for the information to access the Secure FTP server.

3 The config.properties configuration file

The standard location of the configuration file config.properties is /www/HKIRC-sdk/props. If the location of the file is changed, please update the corresponding file path in the sessionObj.java and recompile.

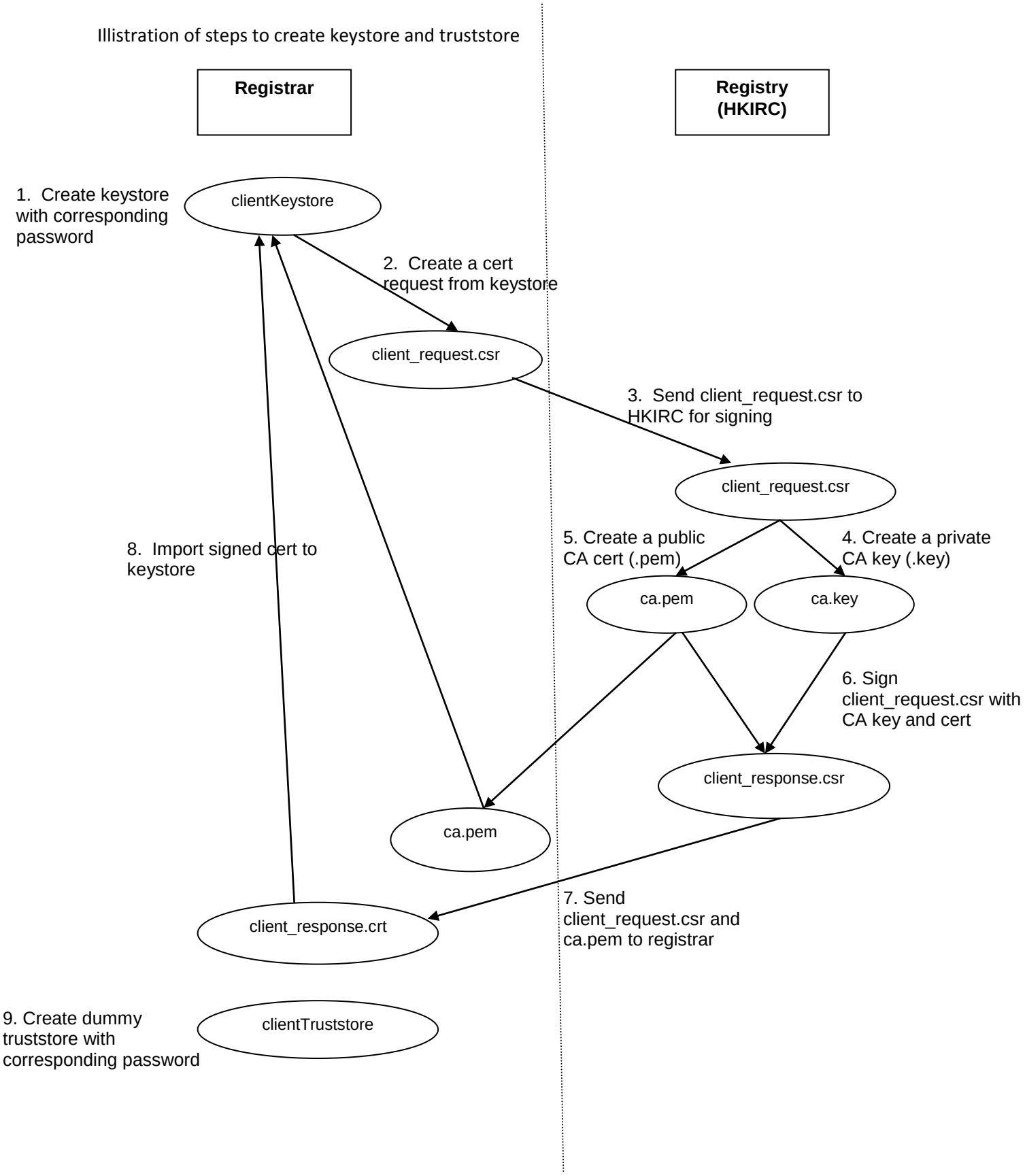
Each registrar should obtain a Registrar ID and password from HKIRC. Please update the ID and password values in the configuration file in order to communicate with the server. Without a valid ID, the EPP API client will not be able to communicate with the EPP server.

```
#
# For JSSE
#
ssl.client.authentication=true
ssl.keymanager.algorithm=SunX509
ssl.keystore.type=JKS
ssl.keystore.provider=SUN
ssl.keystore.format=file
ssl.keystore.name=/www/HKIRC-sdk/props/clientKeystore
ssl.keystore.storepass=xxxxxx
ssl.keystore.keypass=xxxxxx
ssl.trustmanager.algorithm=SunX509
ssl.truststore.type=JKS
ssl.truststore.provider=SUN
ssl.truststore.format=file
ssl.truststore.name=/www/HKIRC-sdk/props/clientTruststore
ssl.truststore.storepass=xxxxxx

#
# Registrar EPP username & password
username=EPP-1234
password=xxxxxx
serverIP=10.0.0.1
serverPort=700
```

Steps to Create EPP Client Keystore and Truststore:

Step	Description	Command	PIC	Output file	Setting in config.properties
1	Create keystore with corresponding password	%JAVA_HOME%/bin/keytool -genkey -keyalg RSA -alias client -keystore clientKeystore	Registrar	clientKeystore	ssl.keystore.name: path of clientKeystore ssl.keystore.storepass: password of keystore ssl.keystore.keypass: password of keystore
2	Create a cert request from keystore	%JAVA_HOME%/bin/keytool -certreq -v -alias client -keystore clientKeystore -file client_request.csr	Registrar	client_request.csr	-
3	Send client_request.csr to HKIRC for signing	-	Registrar	-	-
4	[First time only] Create a private CA key (.key)	openssl genrsa -des3 -out ca.key 1024	Registry (HKIRC)	ca.key	-
5	[First time only] Create a public CA cert (.pem)	openssl req -new -x509 -key ca.key -out ca.pem -days 3600	Registry (HKIRC)	ca.pem	-
6	Sign client_request.csr with CA key and cert	openssl x509 -req -in client_request.csr -CA ca.pem -CAkey ca.key -CAcreateserial -out client_response.crt -extfile /usr/local/openssl/ssl/openssl.cnf -extensions v3_ca -days 3650	Registry (HKIRC)	client_response.crt	/etc/pki/tls/openssl.cnf view content of certificate keytool -v -list -keystore clientKeyStore.dat
7	Send client_response.crt and ca.pem to registrar	-	Registry (HKIRC)	-	-
8	Import signed cert (client_response.csr and ca.pem) to keystore	%JAVA_HOME%/bin/keytool -import -trustcacerts -alias cacert -file ca.pem -keystore clientKeystore %JAVA_HOME%/bin/keytool -import -trustcacerts -alias client -file client_response.crt -keystore clientKeystore	Registrar	-	-
9	Create dummy truststore with corresponding password	%JAVA_HOME%/bin/keytool -genkey -alias dummy -keyalg RSA -keystore clientTruststore	Registrar	clientTruststore	ssl.truststore.name: path of clientTruststore ssl.truststore.storepass: password of truststore
10	Place the keystore and truststore into EPP client props folder (/www/HKIRC-sdk/props)	-	Registrar	-	-


Illistration of steps to create keystore and truststore
Registrar
Registry (HKIRC)
1. Create keystore with corresponding password
clientKeystore
2. Create a cert request from keystore
client_request.csr
3. Send client_request.csr to HKIRC for signing
client_request.csr
5. Create a public CA cert (.pem)
4. Create a private CA key (.key)
ca.pem
ca.key
6. Sign client_request.csr with CA key and cert
client_response.csr
ca.pem
8. Import signed cert to keystore
7. Send client_request.csr and ca.pem to registrar
client_response.crt
9. Create dummy truststore with corresponding password
clientTruststore

EPP Communication

Registrar to Registry communications and responses utilise the Extensible Markup Language (XML) running over the Secure Sockets Layer (SSL) in a form called the Extensible Provisioning Protocol (EPP). It is the Registrar's responsibility to adhere to these protocols to successfully communicate with the Operations & Test Evaluation (OTE) environment for obtaining Accreditation from the Registry. OTE Test guideline will be provided once OTE username and password are distributed.

The Registrars' application client must utilise XML over SSL to send commands to the Registry and utilise an XML parser to interpret the server's responses.

This section is intended to provide users of the Extensible Provisioning Protocol (EPP) Software Development Kit (SDK) an overview of the HKIRC Top Level Domain Product. Below is the description of the TLD interface classes, including the pre-conditions, the post-conditions, the exceptions, the EPP status codes, and sample codes of each of the action methods.

It is assumed that the reader has reviewed the associated EPP specifications and has a general understanding of the EPP concepts. Much of the EPP details are encapsulated in the SDK, but having a solid understanding of the EPP concepts will help in effectively using the SDK.

3.1 Session Management

These methods establish EPP connection with your SSL settings by sending the EPP hello command over and login with the credential provided. EPP server will reply with a Greeting response upon receiving a Hello command.

After 15 minutes of idle time, the Registry will disconnect a Registrar's client connection. It is preferable for Registrars to keep sessions open for extended periods rather than frequently opening and closing the sessions, as the latter can increase the transaction times.

Objects:

EppSessionTcp session = new EppSessionTcp();
EppCommandLogin login = new EppCommandLogin(greeting.getServiceMenu());

Mandatory	Description	Value	Parameter
Yes	HKIRC EPP server host and port	EPP.HKIRC.HK (Port: 700)	EppGreeting greeting = session.connect(host, port);
Yes	EPP login username and password	<as per given in the email>	login.setCreds(new EppCreds(this.username, this.password))

Sample Code to Establish EppSession (Referring sessionObj.java)

```
try {
            String host = "";
            String portStr = "";

            String localpath = "C:"; // Set the root directory. E.g.
"C:", "D:", "E:" and etc. for Windows platform or empty string ("") for
Linux platform
            String configFile = "/www/hknic-
sdk/props/config.properties"; // Set the physical path of
'config.properties' file for HKNIC EPP client application to load the
connection settings.

            configFile = localpath + configFile;
            int port = 0;
            this.session = new EppSessionTcp();

            Properties prop = new Properties();

            // If the application failed to load 'config.properties'
file in the path defined above, it will try to load again without root
directory.
            // In case Linux-platform user did not set empty string as
the root directory above, the client application still able to startup.
            if (!new File(configFile).exists()){
                        localpath = "";
                        configFile = localpath + configFile;
                  }

            // load EPP connection settings from file
                  FileInputStream file = new
FileInputStream(configFile);
                  prop.load(file);

                  // load EPP username from 'config.properties' file
                  if(this.username == null || "".equals(this.username)){
                        this.username = prop.getProperty("username");
                  }

                  // load EPP password from 'config.properties' file
                  if(this.password == null || "".equals(this.password)){
                        this.password = prop.getProperty("password");
                  }

                  // load EPP server IP from 'config.properties' file
                  host = prop.getProperty("serverIP");

                  // load EPP server port from 'config.properties' file
                  portStr = prop.getProperty("serverPort");
                  if (!"".equals(portStr) && portStr!=null){
                        port = Integer.parseInt(portStr);
                  }
```

```
// Initializes the run-time parameters related to an EPP Session.
// This method must be called before the EPP Session is started,
session.connect(host, port), by passing in the configuration file as
parameter.
                  // Load client SSL keys into session
                  if (OTE.equals("0")) {
                this.session.init(configFile);
            }

            /* EPP client start SSL keys (defined in
'config.properties') setting, EPP greeting and SSL handshake with Server
             * Parameters:
             *  - host : EPP server IP
             *  - port : EPP server port
             */
            System.out.println("\n-----------------------------------
-----------------");
            System.out.println("Client start EppGreeting >>> ");

            EppGreeting greeting = this.session.connect(host, port);

            System.out.println("Client get EppGreeting response from
server >>> ");
            System.out.println(greeting);
            System.out.println("-----------------------------------------
---------------\n");

            if( greeting == null )
            {
                        System.out.println("Cannot connect");
                        Exception e = this.session.getException();
                        if( e != null )
                        {
                              e.printStackTrace();
                        }
                        String s = this.session.getMessage();
                        if( s != null )
                        {
                              System.out.println("Message received:\n" +
s);
                        }
                return null;
            }

            this.channel = this.session.getChannel();
/* EPP client start login to EPP Server
             * Parameters:
             *  - login : EppCommandLogin object
             * Pre-requisites:
             *  - setClientTransactionId(id) on EppCommandLogin object
where 'id' is a random generated code [refer to
GenerateRandomID.randomstring()]
             *  - setCreds(new EppCreds(username,password) on
EppCommandLogin object where 'username' and 'password' are EPP login
details
             *  - [ only applicable for changing EPP password ]
setCreds(new EppCreds(username,password,new_password) on EppCommandLogin
object where 'username' and 'password' are EPP login details, and
'new_password' is the new EPP password going to be changed.
*/
```

```
            System.out.println("----------------------------------------
---------------");
            System.out.println("Client start to login >>> ");

            EppCommandLogin login = new
EppCommandLogin(greeting.getServiceMenu());
            String id = getClientId();
            login.setClientTransactionId(id);
            this.channel.setClientId(id);
            if(this.new_password == null ||
"".equals(this.new_password)){
                  login.setCreds(new EppCreds(this.username,
this.password));
            }else{
                  login.setCreds(new EppCreds(this.username,
this.password, this.new_password));
            }

            EppResponse res = this.channel.start(login);

            System.out.println(login);
            System.out.println("\nClient get EppResponse response from
server >>> ");
            System.out.println(res);
            System.out.println("----------------------------------------
---------------\n");

            // Action to handle EPP login failure
            if( res == null )
            {
                  System.out.println("LOGIN error");
                        return null;
            }
            if( ! res.success() )
            {
                  System.out.println(res.getResult());
                        return null;
            }

        }
        catch (Exception ex) {
            ex.printStackTrace();
            System.out.println(ex.getMessage());
        }
        return channel;
```

3.2 Client Interfaces

This portion of the SRS SDK contains client interface classes for each of the EPP Mappings. The interfaces provide mechanisms for creating and modifying hosts, contacts and domains. They also provide the ability to query the status of provisioning requests through the command interface. All of these classes are used for the purpose of provisioning and administering contacts, hosts and domain names in the HKIRC Top-Level Domains. The following sections describe the client interface classes, supporting classes and their respective purposes.

3.2.1 Contact Interface

This interface is used to query, create, update and delete contacts that are associated with domains.

3.2.1.1 Create Contact

This method sends the EPP create contact command.

Pre-Conditions

This method expects that the contact object be populated with the appropriate attributes for the type of contact that is being created (e.g. Organization, Individual). The following list shows fields that are optional and required for the different kinds of contacts

Objects:

```
EppAddress Addr = new EppAddress();
EppContactData ContactData = new EppContactData();
EppContact Contact = new EppContact();
hkExtension hk = new hkExtension();
EppCommandCreate cmd = EppCommand.create(Contact, EppChannel.getClientId());
```

Organization Contact

Fields	Registrant (1)	Administrative (2)	Technical (3)	Billing (4)	Parameter	Maximum Length	Value
Contact Organization Name	✓	✓	✓	✓	ContactData.setOrganization(%value)	150	Hexagon Inc.
Contact Name	✓	✓	✓	✓	ContactData.setName(%value)	60	John
Contact Address	✓	✓	✓	✓	Addr.setStreet(0, %value)	120	12/F, One Kowloon
Contact Address - Postal Code	×	×	×	×	Addr.setPostalCode(%value)	60	90000
Contact Address - Country Code	✓	✓	✓	✓	Addr.setCountryCode(%value)	-	Refer to Appendix
Contact Telephone Number	✓	✓	✓	✓	Contact.setVoice(%value)	32	+852-12312312
Contact Fax Number	×	×	×	×	Contact.setFax(%value)	32	+852-12312312
Contact Email Address	✓	✓	✓	✓	Contact.setEmail(%value)	60	johndoe@email.com
Contact Surname	✓	✓	✓	✓	hk.addExtension(%value, hkExtension.SURNAME)	60	Doe
Contact Category	✓	✓	✓	✓	hk.addExtension(%value, hkExtension. CATEGORY_TYPE)	-	O: For Organizational Domain Contact
Contact Type	✓	✓	✓	✓	hk.addExtension(%value, hkExtension.TYPE)	-	Refer to Appendix
(Cf1) Contact Chinese Company	×	×	×	×	hk.addExtension(%value, hkExtension.CHINESEORG)	60	香港域名註冊有限公司 (UTF-8 encoding)
(Cf2) Contact Document Type	✓	×	×	×	hk.addExtension(%value, hkExtension.DOCTYPE)	-	Refer to Appendix
(Cf3) Contact Document Number	✓	×	×	×	hk.addExtension(%value, hkExtension.DOCNUM)	60	123456
(Cf4) Contact Document Origin Country	✓	×	×	×	hk.addExtension(%value, hkExtension.DOCORIGINCC)	-	Refer to Appendix

(Cf6) Contact Industry Type	✓	×	×	×	hk.addExtension(%value, hkExtension.INDUSTRYTYPE)	-	Refer to Appendix
Other Document	✓ if CF2 is OTHORG	✓ if CF2 is OTHORG	✓ if CF2 is OTHORG	✓ if CF2 is OTHORG	hk.addExtension(%value, hkExtension.OTHERDOC)	60	MASPASS
Mobile Phone Number	×	×	×	×	hk.addExtension(%value, hkExtension.MBNUMBER)	32	+852-12312312

✓	Mandatory	×	Optional

Individual Contact

Fields	Registrant (1)	Technical (3)	Billing (4)	Parameter	Maximum Length	Value
Contact Organization Name	Name + Surname	Name + Surname	Name + Surname	ContactData.setOrganization(%value)	150	John Doe
Contact Name	✓	✓	✓	ContactData.setName(%value)	60	John
Contact Address	✓	✓	✓	Addr.setStreet(0, %value)	120	123, Street X
Contact Address - Postal Code	×	×	×	Addr.setPostalCode(%value)	60	90000
Contact Address - Country Code	✓	✓	✓	Addr.setCountryCode(%value)	-	Refer to Appendix
Contact Telephone Number	✓	✓	✓	Contact.setVoice(%value)	32	+852-12312312
Contact Fax Number	×	×	×	Contact.setFax(%value)	32	+852-12312312
Contact Email Address	✓	✓	✓	Contact.setEmail(%value)	60	johndoe@email.com
Contact Surname	✓	✓	✓	hk.addExtension(%value, hkExtension.SURNAME)	60	Doe
Contact Category	✓	✓	✓	hk.addExtension(%value, hkExtension. CATEGORY_TYPE)	-	I: For Individual Domain
Contact Type	✓	✓	✓	hk.addExtension(%value, hkExtension.TYPE)	-	Refer to Appendix
(Cf1) Contact Chinese Company	×	×	×	hk.addExtension(%value, hkExtension.CHINESEORG)	60	香港域名註冊有限公司 (UTF-8 encoding)
(Cf2) Contact Document Type	✓	×	×	hk.addExtension(%value, hkExtension.DOCTYPE)	-	Refer to Appendix
(Cf3) Contact Document Number	✓	×	×	hk.addExtension(%value, hkExtension.DOCNUM)	60	123456
(Cf4) Contact Document Origin	✓	×	×	hk.addExtension(%value,	-	Refer to

Country				hkExtension.DOCORIGINCC)		Appendix
(Cf5) Is Contact Under Age 18	✓	×	×	hk.addExtension(%value, hkExtension.UNDER18)	-	0: Under age of 18 1: 18 year old or older
Other Document	✓ if CF2 is OTHIDV	✓ if CF2 is OTHIDV	✓ if CF2 is OTHIDV	hk.addExtension(%value, hkExtension.OTHERDOC)	60	MASPASS
Mobile Phone Number	×	×	×	hk.addExtension(%value, hkExtension.MBNUMBER)	32	+852-12312312

✓	Mandatory	×	Optional

Post-Conditions

When completed successfully, an *EppResponseDataCreateContact* object is returned, with the following attributes:

- ID= contact handle ID.

Response Code

1000 ; Command completed successfully
2003 ; Required parameter missing
2004 ; Parameter value range error
2005 ; Parameter value syntax error
2305 ; Object association prohibits operation
2400 ; Command failed

Sample Code

The following example shows the steps of creating a contact through the use of the contact client interface and command send method.
(E.g. to Create an Organisational contact handle)

```
String orgName        = "Hexagon Inc.";
String name           = "John";
String address        = "12/F, One Kowloon";
String postalCode     = "";
String countryCode    = "HK";
String telephone      = "+852-12312312";
String fax            = "+852-12312312";
String email          = "johndoe@email.com";
String surname        = "Doe";
String category       = "O";
String contactType    = "2";
//UTF-8 Encoding
String chineseOrg     = "香港域名註冊有限公司";
String docType        = "BR";
String docNumber      = "123456";
String docOriginCC    = "HK";
String industryType   = "0";
String otherDoc       = "";
String mobilePhone    = "+852-12312312";

// create contact's address info
EppAddress Addr = new EppAddress();
        Addr.setStreet(0, address);       // Address
        Addr.setPostalCode(postalCode);   // postal code
        Addr.setCountryCode(countryCode); // country code

// create contact's person info
EppContactData ContactData = new EppContactData();
        ContactData.setName(name);            // name
        ContactData.setOrganization(orgName); // organization name
        ContactData.setAddress(Addr);         // address

EppContact Contact = new EppContact();
        Contact.setContactDataAscii(ContactData);
        Contact.setVoice(telephone); // telephone
        Contact.setFax(fax);         // fax
        Contact.setEmail(email);     // email

EppCommandCreate cmd = EppCommand.create(Contact,
EppChannel.getClientId());
```

```
// extensions, extra fields required by HKIRC
hkExtension hk = new hkExtension();
      hk.addExtension(surname,hkExtension.SURNAME);
      hk.addExtension(category,hkExtension.CATEGORY_TYPE);
      hk.addExtension(contactType,hkExtension.TYPE);
      hk.addExtension(chineseOrg,hkExtension.CHINESEORG);
      hk.addExtension(docType,hkExtension.DOCTYPE);
      hk.addExtension(docNumber,hkExtension.DOCNUM);
      hk.addExtension(docOriginCC,hkExtension.DOCORIGINCC);
      hk.addExtension(industryType,hkExtension.INDUSTRYTYPE);
      hk.addExtension(otherDoc,hkExtension.OTHERDOC);
      hk.addExtension(mobilePhone,hkExtension.MBNUMBER);

cmd.setEppExtension(hk);

// sends out epp command to server and get response
EppResponse res = EppChannel.send(cmd);
if( res != null ) {
      if( res.success() ) {
            EppResponseDataCreateContact res_data =
            (EppResponseDataCreateContact) res.getResponseData();
            if( res_data != null ) {
                  System.out.println("Contact Created: Contact ID " +
            res_data.getId());
            } else {
                  // throw handle exception
            }
      } else {
          printErrors(res.getResult());
      }
} else {
      // throw handle exception
}
```

3.2.1.2 Update Contact

This method sends the EPP update contact command to modify the contact information.

Pre-Conditions

This method expects that the contact object be populated with the appropriate attributes for single contact identifier of the contact that is being updated. In addition, all of the attributes originally set during the creation of contact must be populated. If an attribute is left null, it will be set to null in the registry as well. This also depends on what kind of contact is being updated (e.g. organization or individual). The following list shows fields that are optional and required for the different kinds of contacts

Objects:

```
EppAddress Addr = new EppAddress();
EppContactData ContactData = new EppContactData();
EppContact Contact = new EppContact();
hkExtension hk = new hkExtension();
EppCommandUpdateContact contact = (EppCommandUpdateContact)
EppCommand.update(EppObject.CONTACT, contactid, EppChannel.getClientId());
```

Organization Contact

Fields	Registrant (1)	Administrative (2)	Technical (3)	Billing (4)	Parameter	Maximum Length	Value
Contact Handler Identifier	✓	✓	✓	✓	(EppCommandUpdateContact)EppCommand.update(EppObject.CONTACT, %value, EppChannel.getClientId());	-	HK1713004T
Contact Name	✓	✓	✓	✓	ContactData.setName(%value)	60	John
Contact Address	✓	✓	✓	✓	Addr.setStreet(0, %value)	120	12/F, One Kowloon
Contact Address - Postal Code	×	×	×	×	Addr.setPostalCode(%value)	60	90000
Contact Address - Country Code	✓	✓	✓	✓	Addr.setCountryCode(%value)	-	Refer to Appendix
Contact Telephone Number	✓	✓	✓	✓	Contact.setVoice(%value)	32	+852-12312312
Contact Fax Number	×	×	×	×	Contact.setFax(%value)	32	+852-12312312
Contact Email Address	✓	✓	✓	✓	Contact.setEmail(%value)	60	johndoe@email.com
Contact Surname	✓	✓	✓	✓	hk.addExtension(%value, hkExtension.SURNAME)	60	Doe
(Cf5) Is Contact Under Age 18	×	×	×	×	hk.addExtension(%value, hkExtension.UNDER18)	-	0: Under age of 18 1: 18 year old or older
(Cf6) Contact Industry Type	✓	×	×	×	hk.addExtension(%value, hkExtension.INDUSTRYTYPE)	-	Refer to Appendix
Mobile Phone Number	×	×	×	×	hk.addExtension(%value, hkExtension.MBNUMBER)	32	+852-12312312

✓	Mandatory	×	Optional

Individual Contact

Fields	Registrant (1)	Technical (3)	Billing (4)	Parameter	Maximum Length	Value
Contact Handler Identifier	✓	✓	✓	(EppCommandUpdateContact)EppCommand.update(EppObject.CONTACT, %value, EppChannel.getClientId())	-	HK1713004T
Contact Name	✓	✓	✓	ContactData.setName(%value)	60	John
Contact Address	✓	✓	✓	Addr.setStreet(0, %value)	120	123, Street X
Contact Address - Postal Code	×	×	×	Addr.setPostalCode(%value)	60	90000
Contact Address - Country Code	✓	✓	✓	Addr.setCountryCode(%value)	-	HK
Contact Telephone Number	✓	✓	✓	Contact.setVoice(%value)	32	+852-12312312
Contact Fax Number	×	×	×	Contact.setFax(%value)	32	+852-12312312
Contact Email Address	✓	✓	✓	Contact.setEmail(%value)	60	johndoe@email.com
Contact Surname	✓	✓	✓	hk.addExtension(%value, hkExtension.SURNAME)	60	Doe
(Cf5) Is Contact Under Age 18	✓	×	×	hk.addExtension(%value, hkExtension.UNDER18)	-	0: Under age of 18 1: 18 year old or older
(Cf6) Contact Industry Type	×	×	×	hk.addExtension(%value, hkExtension.INDUSTRYTYPE)	-	Refer to Appendix
Mobile Phone Number	×	×	×	hk.addExtension(%value, hkExtension.MBNUMBER)	32	+852-12312312

✓	Mandatory	×	Optional

Post-Conditions

- When completed successfully, an *EppResponse* object is returned.

Response Code

1000 ; Command completed successfully
2003 ; Required parameter missing
2004 ; Parameter value range error
2005 ; Parameter value syntax error
2303 ; Object does not exists
2305 ; Object association prohibits operation
2201 ; Authorization error
2400 ; Command failed

Sample Code

The following example shows the steps of updating a contact through the use of the contact client interface and command send method. (E.g. to Update an Organizational contact handle)

```
String id             = "HK1713004T";
String name           = "John";
String address        = "12/F, One Kowloon";
String postalCode     = "";
String countryCode    = "HK";
String telephone      = "+852-12312312";
String fax            = "+852-12312312";
String email          = "johndoe@email.com";
String surname        = "Doe";
String isUnder18      = "0";
String industryType   = "0";
String mobilePhone    = "+852-12312312";

// create contact's address info
EppAddress Addr = new EppAddress();
        Addr.setStreet(0, address);       // address
        Addr.setPostalCode(postalCode);   // postal code
        Addr.setCountryCode(countryCode); // country code

// create contact's person info
EppContactData ContactData = new EppContactData();
        ContactData.setAddress(Addr); //EppAddress Object
        ContactData.setName(name);    //Name

EppCommandUpdateContact contact =
(EppCommandUpdateContact)EppCommand.update(EppObject.CONTACT, contactid,
EppChannel.getClientId());
        contact.setNewAscii(ContactData);
        contact.setNewVoice(telephone); // telephone
        contact.setNewFax(fax);         // fax
        contact.setNewEmail(email);     // email

// extensions, extra fields required by HKIRC
hkExtension hk = new hkExtension();
        hk.addExtension(surname,hkExtension.SURNAME);
        hk.addExtension(isUnder18,hkExtension.UNDER18);
        hk.addExtension(industryType,hkExtension.IDUSTRYTYPE);
        hk.addExtension(mobilePhone,hkExtension.MBNUMBER);

cmd.setEppExtension(hk);
```

```
// sends epp command to server and get response
EppResponse res = EppChannel.send(cmd);

if( res != null ) {
      if( res.success() ) {
            System.out.println("Contact Updated Successfully);
      } else {
            printErrors(res.getResult());
      }
} else {
      // throw handle exception
}
```

3.2.1.3 Delete Contact

This method sends the EPP delete contact command to delete a contact handle.

Pre-Conditions

This method expects that the contact object be populated with the appropriate attributes for single contact identifier of the contact that is being deleted. In addition, this method requires that no domains are associated with the contact prior to deletion. If there are domains associated with the contact the deletion will fail. The following list shows field that is required for the method call.

Objects:
EppCommandDelete cmd = EppCommand.delete(EppObject.Contact, contactid, EppChannel.getClientId());

Organizational Contact

Mandatory	Description	Value	Parameter
Yes	Contact Handle Identifier	HKxxxxxxxT	EppCommandDelete cmd = EppCommand.delete(EppObject.Contact, %value, EppChannel.getClientId());

Individual Contact

Mandatory	Description	Value	Parameter
Yes	Contact Handle Identifier	HKxxxxxxxxT	EppCommandDelete cmd = EppCommand.delete(EppObject.Contact, %value, EppChannel.getClientId());

Post-Conditions

- When completed successfully, an *EppResponse* object is returned.

Response Code

1000 ; Command completed successfully
2003 ; Required parameter missing
2303 ; Object does not exists
2305 ; Object association prohibits operation
2201 ; Authorization error
2308 ; Data management policy violation
2400 ; Command failed

Sample Code

The following example shows the steps of deleting a contact through the use of the contact client interface and command send method. (E.g. to delete an Organizational contact handle)

```
String contactid=”HK1000001T”;

EppCommandDelete cmd = EppCommand.delete(EppObject.Contact, contactid,
EppChannel.getClientId());

EppResponse res = EppChannel.send(cmd);

if( res != null ) {
      if( res.success() ) {
            System.out.println("Contact Deleted”);
      } else {
           printErrors(res.getResult());
      }
} else {
      // throw handle exception
}
```

3.2.1.4 Info Contact

This method sends the EPP info contact command to retrieve contact handle information.

Pre-Conditions

This method expects that the contact object be populated with the appropriate attributes for single contact identifier of the contact that is being queried. The following list shows field that is required for the method called.

Objects:
EppCommandInfo cmd = EppCommand.info(EppObject.Contact, contactid, EppChannel.getClientId());

Organisational Contact

Mandatory	Description	Value	Parameter
Yes	Contact Handle Identifier	HKxxxxxxxT	EppCommandInfo cmd = EppCommand.info(EppObject.Contact, %value, EppChannel.getClientId());

Individual Contact

Mandatory	Description	Value	Parameter
Yes	Contact Handle Identifier	HKxxxxxxxT	EppCommandInfo cmd = EppCommand.info(EppObject.Contact, %value, EppChannel.getClientId());

Post-Conditions

- When completed successfully, *EppResponseDataInfo, EppContact* object is returned.

Response Code

1000 ; Command completed successfully
2003 ; Required parameter missing
2303 ; Object does not exists
2201 ; Authorization error
2400 ; Command failed

Sample Code

The following example shows the steps of querying a contact through the use of the contact client interface and command send method. (E.g. to query an Organisational contact handle)

```
String contactid="HKNIC-ORG1000001";

EppCommandInfo cmd = EppCommand.info(EppObject.Contact, contactid,
EppChannel.getClientId());

EppResponse res = EppChannel.send(cmd);

if( res != null ) {
      if( res.success() ) {
            EppResponseDataInfo res_data =
            (EppResponseDataInfo) res.getResponseData();

            if( res_data != null ) {
                  EppContact Contact = (EppContact)
res_data.getObject();
                  //retrieve Contact attribute here
            } else {
                  // throw handle exception
            }
      } else {
          printErrors(res.getResult());
      }
} else {
      // throw handle exception
}
```

3.2.2 Host Interface

This interface is used to query, check, create, update and delete hosts that are associated with domains.

3.2.2.1 Check Host

This method sends the EPP check host command to check the allowable flag for one or more hosts.

Pre-Conditions

This method expects that the host object be populated with one or more host names. The following list shows field that is required for the method.

Objects

EppCommandCheck cmd = EppCommand.check(EppObject.HOST, EppChannel.getClientId());

Mandatory	Description	Value	Parameter
Yes	Host Name	ns1.myhost.com.hk	cmd.add(%value)

Post-Conditions

When completed successfully, an *EppResponseDataCheck* object is returned, with the following attributes:

- isAvailable= the check results are returned in a collection containing one or more hosts.

Response Code

1000; command completed successfully
2001; command syntax error

Sample Code

The following example shows the steps of checking one or more hosts through the use of the host client interface and command send method.

```
String hostname=”ns.myhost.com.hk”;

EppCommandCheck cmd = EppCommand.check(EppObject.HOST,
EppChannel.getClientId());

cmd.add(hostname);

EppResponse res = EppChannel.send(cmd);

if( res != null ) {
      if( res.success() ) {
            EppResponseDataCheck res_data =
            (EppResponseDataCheck) res.getResponseData();

            if( res_data != null ) {
               if (res_data.isAvailable(hostname))
               {
                  System.out.println("HostCheck: Host " + hostname + "
                  is available");
               } else {
                  System.out.println("HostCheck: Host " + hostname + "
                  is not available");
               }
            } else {
                  // throw handle exception
            }
      } else {
          printErrors(res.getResult());
      }
} else {
      // throw handle exception
}
```

3.2.2.2 Create Host

This method sends the EPP create host command.

Pre-Conditions

This method expects that the host object be populated with one or more host names. The following list shows fields that are required for the method.

Objects

EppHost Host = new EppHost();
EppCommandCreate cmd = EppCommand.create(Host, EppChannel.getClientId());

Mandatory	Description	Value	Parameter
Yes	Host Name	ns1.myhost.com.hk	Host.setName(%value);
Yes	Host IP	202.188.101.11	Host.addAddress(new EppIpAddress(%value, "v4"));
Yes	Host IP Type	v4 or v6	Host.addAddress(new EppIpAddress("202.166.12.18", %value));

Post-Conditions

When completed successfully, an *EppResponseDataCreateHost* object is returned, with the following attributes:

- Name = the host name that was successfully created.

Response Code

1000 ; Command completed successfully
2003 ; Required parameter missing
2005 ; Parameter value syntax error
2201 ; Authorization error
2302 ; Object exists
2303 ; Object does not exists
2305 ; Object association prohibits operation
2306 ; Parameter value policy error
2400 ; Command failed

Sample Code

The following example shows the steps of performing create of hosts through the use of the host client interface and command send method.

```
EppHost Host = new EppHost();

String hostname=”ns.myhost.com.hk”;

Host.setName(hostname);

Host.addAddress(new EppIpAddress("202.166.12.18", "v4"));

Host.addAddress(new EppIpAddress("202.166.12.19", "v4"));

EppCommandCreate cmd = EppCommand.create(Host,
EppChannel.getClientId());

EppResponse res = EppChannel.send(cmd);

if( res != null ) {
      if( res.success() ) {
            EppResponseDataCreateHost res_data =
            (EppResponseDataCreateHost) res.getResponseData();

            if( res_data != null ) {
                  System.out.println(“Host created: ” +
                  res_data.getName());
            } else {
                  // throw handle exception
            }
      } else {
          printErrors(res.getResult());
      }
} else {
      // throw handle exception
}
```

3.2.2.3 Update Host

This method sends the EPP update host command to modify host name or host IP.

Pre-Conditions

This method expected that the host object be populated with the name of the host to be updated and the Ipv4 or Ipv6 address to change. In addition, this method allows change of the host name. The following list shows fields that are required for the method.

Objects

EppCommandUpdateHost cmd = EppCommand.update(EppObject.HOST, hostname, EppChannel.getClientId());

Mandatory	Description	Value	Parameter
Yes	Host Name	ns1.myhost.com.hk	EppCommandUpdateHost cmd = EppCommand.update(EppObject.HOST, %value, EppChannel.getClientId());

Yes	Host IP (add or remove)	202.188.101.11	cmd.addAddress(new EppIpAddress(%value, "v4"));
			
			or
			
			cmd.removeAddress(new EppIpAddress(%value, "v4"));

Yes	Host IP Type	v4 or v6	cmd.addAddress(new EppIpAddress(ip, %value));
			
			or
			
			cmd.removeAddress(new EppIpAddress(ip, %value));

Post-Conditions

When completed successfully, a standard *EppResponse* object is returned.

Response Code

1000 ; Command completed successfully
2003 ; Required parameter missing
2005 ; Parameter value syntax error
2201 ; Authorization error
2302 ; Object exists
2303 ; Object does not exists
2305 ; Object association prohibits operation
2306 ; Parameter value policy error

2400 ; Command failed

Sample Code

The following example shows the steps of performing an update of hosts through the use of the host client interface and command send method.

```
String hostname=”ns.myhost.com.hk”;

EppCommandUpdateHost cmd = EppCommand.update(EppObject.HOST, hostname,
EppChannel.getClientId());

// change of host ip address
cmd.addAddress(new EppIpAddress("202.111.44.15", "v4"));
cmd.removeAddress(new EppIpAddress("202.111.44.14", "v4"));

// change of host name
cmd.setNewName(“ns1.myhost.com.hk”);

EppResponse res = EppChannel.send(cmd);

if( res != null ) {
      if( res.success() ) {
            System.out.println("Host Updated”);
      } else {
           printErrors(res.getResult());
      }
} else {
      // throw handle exception
}
```

3.2.2.4 Delete Host

This method sends the EPP delete host command.

Pre-Conditions

This method expects that the host object is populated with the appropriate attributes for single host name of the host that is being deleted. In addition, this method requires that no domains are associated with the host prior to deletion. If there are domains associated with the host, the deletion will fail. The following list shows field that is required for the method called.

Objects

EppCommandDelete cmd = EppCommand.delete(EppObject.HOST, hostname, EppChannel.getClientId());

Mandatory	Description	Value	Parameter
Yes	Host Name	ns.myhost.com.hk	EppCommandDelete cmd = EppCommand.delete(EppObject.HOST, %value, EppChannel.getClientId());

Post-Conditions

- When completed successfully, an *EppResponse* object is returned.

Response Code

1000 ; Command completed successfully
2003 ; Required parameter missing
2303 ; Object does not exists
2305 ; Object association prohibits operation
2308 ; Data management policy violation
2400 ; Command failed

Sample Code

The following example shows the steps of deleting a host through the use of the host client interface and command send method.

```
String hostname=”ns.myhost.com.hk”;

EppCommandDelete cmd = EppCommand.delete(EppObject.HOST, hostname,
EppChannel.getClientId());

EppResponse res = EppChannel.send(cmd);

if( res != null ) {
      if( res.success() ) {
            System.out.println("Host Deleted”);
      } else {
           printErrors(res.getResult());
      }
} else {
      // throw handle exception
}
```

3.2.2.5 Info Host

This method sends the EPP info host command to retrieve host information.

Pre-Conditions

This method expects that the host object be populated with the appropriate attributes for single host name of the host that is being queried. The following list shows the field required for the method.

Objects

EppCommandInfo cmd = EppCommand.info(EppObject.HOST, hostname, EppChannel.getClientId());

Mandatory	Description	Value	Parameter
Yes	Host Name	ns.myhost.com.hk	EppCommandInfo cmd = EppCommand.info(EppObject.HOST, %value, EppChannel.getClientId());

Post-Conditions

- On success, *EppResponseDataInfo, EppHost* object is returned.

Response Code

1000 ; Command completed successfully
2005 ; Parameter value syntax error
2303 ; Object does not exists
2400 ; Command failed

Sample Code

The following example shows the steps of querying a host through the use of the host client interface and command send method.

```
String hostname="ns.myhost.com.hk";

EppCommandInfo cmd = EppCommand.info(EppObject.HOST, hostname,
EppChannel.getClientId());

EppResponse res = EppChannel.send(cmd);

if( res != null ) {
      if( res.success() ) {
            EppResponseDataInfo res_data =
            (EppResponseDataInfo) res.getResponseData();

            if( res_data != null ) {
                  EppHost Host = (EppHost) res_data.getObject();
                  //retrieve Host attribute here
            } else {
                  // throw handle exception
            }
      } else {
           printErrors(res.getResult());
      }
} else {
      // throw handle exception
}
```

3.2.3 Domain Interface

This interface is used to query, check, create, renew, transfer, update and delete domain names.

3.2.3.1 Check Domain

This method sends the EPP check domain command to check the allowable flag for one or more domains.

Pre-Conditions

This method expects that the domain object be populated with one or more domain names. The following list shows the required field for the method.

Objects

EppCommandCheck cmd = EppCommand.check(EppObject.DOMAIN, EppChannel.getClientId());

Mandatory	Description	Value	Parameter
Yes	Domain Name	mydomain.com.hk	cmd.add(%value)

Post-Conditions

When completed successfully, an *EppResponseDataCheck* object is returned, with the following attributes:

- isAvailable= the check results are returned in a collection containing one or more hosts.

Response Code

1000; command completed successfully
2001; command syntax error

Sample Code

The following example shows the steps of checking one or more domains through the use of the domain client interface and command send method.

```
String domain=”mydomain.com.hk”;

EppCommandCheck cmd = EppCommand.check(EppObject.DOMAIN,
EppChannel.getClientId());

cmd.add(domain);

EppResponse res = EppChannel.send(cmd);

if( res != null ) {
      if( res.success() ) {
            EppResponseDataCheck res_data =
            (EppResponseDataCheck) res.getResponseData();

            if( res_data != null ) {
               if (res_data.isAvailable(domain))
               {
                  System.out.println("DomainCheck: Domain Name " +
                  domain + " is available");
               } else {
                  System.out.println("DomainCheck: Domain Name " +
                  domain + " is not available");
               }
            } else {
                  // throw handle exception
            }
      } else {
           printErrors(res.getResult());
      }
} else {
      // throw handle exception
}
```

3.2.3.2 Create Domain

This method sends the EPP create domain command.

Pre-Conditions

This method requires that several attributes be set prior to execution. The following list shows fields that are required for the method. (Important Note: please avoid reusing the contact Identifier when create a new domain)

Objects

```
EppDomain domain = new EppDomain("mydomain.com.hk");
EppCommandCreate cmd = EppCommand.create(domain,
EppChannel.getClientId());
hkExtension hk = new hkExtension();
```

Mandatory	Description	Value	Parameter
Yes	Domain Name	mydomain.com.hk	EppDomain domain = new EppDomain(%value);
Yes	Registrant Contact Identifier	HKxxxxxxxT	domain.setRegistrant(%value)
Yes, no if Domain Category is 'O'	Administrative Contact Identifier	HKxxxxxxxT	domain.addContact(%value, EppDomain.CONTACT_TYPE_ADMIN);
Yes	Technical Contact Identifier	HKxxxxxxxT	domain.addContact(%value, EppDomain.CONTACT_TYPE_TECH);
Yes	Billing Contact Identifier	HKxxxxxxxT	domain.addContact(%value, EppDomain.CONTACT_TYPE_BILLING);
Yes	Domain Periods	1	domain.setPeriod(new EppPeriod(%value, EppPeriod.UNIT_YEAR));
Yes	Name Server (Max 13)	ns.myhost.com.hk	domain.addNameServer(%value)
Yes	Domain Category	I or O I – Individual O - Organization	hk.addExtension(%value, *hkExtension.CATEGORY_TYPE*);
No	Add Domain's Reseller Information	Reseller Info	hk.addExtension(%value, hkExtension.RESELLER_INFO);
No	Promotion Code	xxxxxxxxxxxxxxx	domain.setPromotion(%value)

Post-Conditions

When completed successfully, an *EppResponseDataCreateDomain* object is returned, with the following attributes:

- Domain Name = the domain name that was successfully created.
- Date Create = domain name creation timestamp.
- Date Expire = domain name expiration timestamp.

Response Code:

1001 ; command completed successfully; action pending
2001 ; command syntax error
2003 ; required parameter missing
2004 ; parameter value range error
2005 ; parameter value syntax error
2104 ; billing failure
2201 ; authorization error
2302 ; object exists
2303 ; object does not exists
2306 ; parameter value policy error
2400 ; command failed

Sample Code

The following example shows the steps of performing create of domain through the use of the domain client interface and command send method.

```
// Set domain information
EppDomain domain = new EppDomain("mydomain.com.hk"); //Domain Name
      // Registrant Contact Handle ID
      domain.setRegistrant("HK1000001T");
      // Administrative Contact Handle ID
      domain.addContact("HK1000002T",
                        EppDomain.CONTACT_TYPE_ADMIN);
      // Technical Contact Handle ID
      domain.addContact("HK1000003T",
                        EppDomain.CONTACT_TYPE_TECH);
      // Billing Contact Handle ID
      domain.addContact("HK1000004T",
                        EppDomain.CONTACT_TYPE_BILLING);
      // Domain Term
      domain.setPeriod(new EppPeriod(1, EppPeriod.UNIT_YEAR));

      // Domain Name Server
      domain.addNameServer("ns1.hknic.com.hk");
      // Domain Name Server
      domain.addNameServer("ns2.hknic.com.hk");
      domain.addPromotion("example-promotion-code");

      EppCommandCreate cmd = EppCommand.create(domain,
                        EppChannel.getClientId());

      // Set Domain Language Tag
      hkExtension hk = new hkExtension();
      hk.addExtension("O", hkExtension.CATEGORY_TYPE);
      if(resellerInformation != null) {
        hk.addExtension("Reseller ABC Details", hkExtension.RESELLER_INFO );
      }
      cmd.setEppExtension(hk);

EppResponse res = EppChannel.send(cmd);

if( res != null ) {
      if( res.success() ) {
            EppResponseDataCreateDomain res_data =
            (EppResponseDataCreateDomain) res.getResponseData();
            if( res_data != null )
            {
                  String domainname = res_data.getName();
                  if (res_data.getDateCreated()!=null){
                    Date crDate = res_data.getDateCreated().getTime();
                  }
                  if (res_data.getDateExpired()!=null){
                    Date exDate = res_data.getDateExpired().getTime();
                  }
            }else{
                  // throw handle exception
            }
      } else {
            printErrors(res.getResult());
      }
} else {
      // throw handle exception
}
```

3.2.3.3 Renew Domain

This method sends the EPP renew domain command.

Pre-Conditions

This method requires that several attributes be set prior to execution. The following list shows fields that are required for the method.

Objects

EppCommandRenewDomain cmd = (EppCommandRenewDomain) EppCommand.renew(EppObject.DOMAIN, domainname, EppChannel.getClientId());
hkExtension hk = new hkExtension();

Mandatory	Description	Value	Parameter
Yes	Domain Name	mydomain.com.hk	EppCommandRenewDomain cmd = (EppCommandRenewDomain) EppCommand.renew(EppObject.DOMAIN, %value, EppChannel.getClientId());
Yes	Term	1	cmd.setPeriod(new EppPeriod(1, EppPeriod.UNIT_YEAR));
Yes	Domain current expiration date	Calendar Date (e.g. 2008-01-01)	cmd.setCurrentExpireDate(%value);
Yes if bundled	Bundle Domain Name	mybundledomain.com.hk	hk.domainExtension(%value, hkExtension.BUNDLE_DOMAIN_NAME)

Post-Conditions

When completed successfully, an *EppResponseDataRenewDomain* object is returned, with the following attributes:

- Domain Name = the domain name that was successfully renewed.
- New Domain Expire Date = the new date that the domain is due to be renewed.

Response Code:

1000 ; command completed successfully
2001 ; command syntax error
2003 ; required parameter missing
2004 ; parameter value range error
2005 ; parameter value syntax error
2104 ; billing failure
2201 ; authorization error
2300 ; object pending transfer
2303 ; object does not exists
2304 ; object status prohibits operation
2306 ; parameter value policy error

2400 ; command failed

Sample Code

The following example shows the steps of performing renew of domain through the use of the domain client interface and command send method.

```
String domainname="mydomain.com.hk";
Calendar cl = Calendar.getInstance();
cl.setTime(CurExpireDate); //current domain expiration date

EppCommandRenewDomain cmd =
(EppCommandRenewDomain)EppCommand.renew(EppObject.DOMAIN, domainname,
EppChannel.getClientId());
// Renew Term
cmd.setPeriod(new EppPeriod(1, EppPeriod.UNIT_YEAR));
// Domain's Current Expiration Date
cmd.setCurrentExpireDate(cl);

EppResponse res = EppChannel.send(cmd);

if( res != null ) {
      if( res.success() ) {
            EppResponseDataRenewDomain res_data =
(EppResponseDataRenewDomain) res.getResponseData();
            if( res_data != null )
            {
                  System.out.println("Domain Name: " +
res_data.getName());
                  if (res_data.getDateExpired()!=null)
                        System.out.println("Expiration Date : " +
res_data.getDateExpired().getTime());

            }else{
                  // throw handle exception
            }
      } else {
            printErrors(res.getResult());
      }
} else {
      // throw handle exception
}
```

3.2.3.4 Delete Domain

This method sends the EPP delete domain command.

Pre-Conditions

This method expects that the domain object be populated with the unique identifier of the domain to be deleted. The following list shows fields that are required for the method.

Objects

EppCommandDelete cmd = EppCommand.delete(EppObject.DOMAIN, domainname, EppChannel.getClientId());
hkExtension hk = new hkExtension();

Mandatory	Description	Value	Parameter
Yes	Domain Name	mydomain.com.hk	EppCommandDelete cmd = EppCommand.delete(EppObject.DOMAIN, %value, EppChannel.getClientId());
Yes if bundled	Bundle Domain Name	mybundledomain.com.hk	hk.domainExtension(%value, hkExtension.BUNDLE_DOMAIN_NAME)

Post-Conditions

When completed successfully, an *EppResponse* object is returned, with the following attributes:

Response Code:

1001 ; command completed successfully
2001 ; command syntax error
2003 ; required parameter missing
2201 ; authorization error
2300 ; object pending transfer
2303 ; object does not exists
2304 ; object status prohibits operation
2306 ; parameter value policy error
2400 ; command failed

Sample Code

The following example shows the steps of performing delete of domain through the use of the domain client interface and command send method.

```
String domainName    = "mydomain.com.hk";
String bunDomainName = "mybundledomain.com.hk";

EppCommandDelete cmd = EppCommand.delete(EppObject.DOMAIN, domainName,
EppChannel.getClientId());

// extensions, extra fields required by HKIRC
hkExtension hk = new hkExtension();

hk.domainExtension(bunDomainName, hkExtension.BUNDLE_DOMAIN_NAME);

cmd.setEppExtension(hk);

EppResponse res = EppChannel.send(cmd);

if( res != null ) {
      if( res.success() ) {
            System.out.println("Domain Deleted");
      } else {
            printErrors(res.getResult());
      }
} else {
      // throw handle exception
```

3.2.3.5 Update Domain

This method sends the EPP update domain command.

Pre-Conditions

This method expects that the domain object be populated with the unique identifier of the domain to be updated and the attribute to be changed. In addition, this method does not allow the change of domain registrant contact identifier, as it would be considered an ownership transfer. The following list shows fields that are optional or required for the method.

Objects

EppCommandUpdateDomain cmd = (EppCommandUpdateDomain) EppCommand.update(EppObject.DOMAIN, domainname, channel.getClientId());

Mandatory	Description	Value	Parameter
Yes	Domain Name	mydomain.com.hk	EppCommandUpdateDomain cmd = (EppCommandUpdateDomain) EppCommand.update(EppObject.DOMAIN, %value, channel.getClientId());
No	Add Name Server	ns3.myhost.com.hk	cmd.addNameServer(%value);
No	Remove Name Server	ns2.myhost.com.hk	cmd.removeNameServer(%value);
No	Add Status	clientHold	cmd.addStatus(%value);
No	Remove Status	clientUpdateProhibited	cmd.removeStatus(%value);
No	Add Update Prohibited Exemption of Modify Domain (MDN) and Modify NS (MNS) or both	MDN,MNS	hk.addExtension(%value, hkExtension.CLIENT_UPDATE_PROHIBITED_EXEMPTION);

Domain Status

clientHold = Hold / Suspend
clientUpdateProhibited = Update Prohibited
clientTransferProhibited = Transfer Prohibited
clientRenewProhibited = Renew Prohibited
clientDeleteProhibited = Delete Prohibited

Update Prohibited Exemption Value

MDN = Modify Domain (MDN)
MNS = Modify NS (MNS)

(Important Note: By default MDN and MNS are update prohibited if no Update Prohibited Exemption value given)

Post-Conditions

When completed successfully, a standard *EppResponse* object is returned.

Response Code

1000 ; command completed successfully
2001 ; command syntax error
2003 ; required parameter missing
2004 ; parameter value range error
2005 ; parameter value syntax error
2102 ; unimplemented option
2201 ; authorization error
2300 ; object pending transfer
2303 ; object does not exists
2304 ; object status prohibits operation
2306 ; parameter value policy error
2400 ; command failed

Sample Code

The following example shows the steps of performing an update of domain.

```
EppCommandUpdateDomain cmd = (EppCommandUpdateDomain)
EppCommand.update(EppObject.DOMAIN, "mydomain.com.hk",
channel.getClientId());


// Update Name Server
      cmd.addNameServer("ns3.hknic.com.hk");
      cmd.removeNameServer("ns2.hknic.com.hk");

//// Update Domain's Client Key Status
// Client Key
// clientDeleteProhibited
// clientHold
// clientRenewProhibited
// clientTransferProhibited
// clientUpdateProhibited

// HoldKey (Required if client key set to "clienthold")
// 1 : Registrar Request
// 2 : Court order
// 3 : Registrant Request
// 4 : Breach of Contract

//// Update Prohibited Exemption Value
// MDN
// MNS
      cmd.addStatus("clientHold");
      cmd.removeStatus("clientUpdateProhibited");

      hkExtension hk = new hkExtension();
      hk.domainExtension("bundledomain.com.hk",
hkExtension.BUNDLE_DOMAIN_NAME);
      hk.addExtension("MDN,MNS",
hkExtension.CLIENT_UPDATE_PROHIBITED_EXEMPTION);
      cmd.setEppExtension(hk);

EppResponse res = EppChannel.send(cmd);
if( res != null ) {
      if( res.success() ) {
            System.out.println("Domain Updated Successfully");
      } else {
            printErrors(res.getResult());
      }
} else {
      // throw handle exception
}
```

3.2.3.6 Transfer of Holding Right / Modify Domain Name Contact

This method sends the EPP update domain command. Note that the followings are the same for both transfer of holding right and modify domain name contact but modify domain name contact **does not contain** registrant contact identifier.

Pre-Conditions

This method expects that the domain object be populated with the unique identifier of the domain to be updated and the registrant contact identifier. The following list shows fields that are required for the method.

Objects

EppCommandUpdateDomain cmd = (EppCommandUpdateDomain) EppCommand.update(EppObject.DOMAIN, domainname, channel.getClientId());
hkExtension hk = new hkExtension();

Mandatory	Description	Value	Parameter
Yes	Domain Name	mydomain.com.hk	EppCommandUpdateDomain cmd = (EppCommandUpdateDomain) EppCommand.update(EppObject.DOMAIN, %value, channel.getClientId());
Yes (For TNR only)	New Registrant Contact Identifier	HK1713634T	cmd.setNewRegistrant(%value);
No	Add Contact Identifier	HK1000009T	cmd.addContact(new EppContactType(%value, EppDomain.CONTACT_TYPE_ADMIN));
No	Remove Contact Identifier	HK1000003T	cmd.removeContact(new EppContactType(%value, EppDomain.CONTACT_TYPE_ADMIN));
Yes (for bundled TNR)	Bundle Domain Name	mybundledomain.com.hk	hk.addExtension(%value, hkExtension.BUNDLE_DOMAIN_NAME);
No	Add, Modify, Removed Domain's Reseller Information	Reseller Info	hk.addExtension(%value, hkExtension.RESELLER_INFO);

* Each Contact Identifier fields are not mandatory but must fill in at least one.

Post-Conditions

On success, a standard *EppResponse* object is returned.

Response Code

1001 ; command completed successfully; action pending
2001 ; command syntax error
2003 ; required parameter missing
2004 ; parameter value range error
2005 ; parameter value syntax error
2102 ; unimplemented option

2201 ; authorization error
2300 ; object pending transfer
2303 ; object does not exists
2304 ; object status prohibits operation
2306 ; parameter value policy error
2400 ; command failed

Sample Code

The following example shows the steps of performing a holding right transfer of domain through the use of the domain client interface and command send method.

```
EppCommandUpdateDomain cmd = (EppCommandUpdateDomain)
EppCommand.update(EppObject.DOMAIN, "mydomain.com.hk",
channel.getClientId());

// update Registrant contact handle id
cmd.setNewRegistrant("HK1713634");

// Update adminitrative contact handle id
      cmd.addContact(new EppContactType("HK1000008T",
                  EppDomain.CONTACT_TYPE_ADMIN));
      cmd.removeContact(new EppContactType("HK1000002T",
                  EppDomain.CONTACT_TYPE_ADMIN));

// Update technical contact handle id
      cmd.addContact(new EppContactType("HK1000009T",
                  EppDomain.CONTACT_TYPE_TECH));
      cmd.removeContact(new EppContactType("HK1000003T",
                  EppDomain.CONTACT_TYPE_TECH));

// Update billing contact handle id
      cmd.addContact(new EppContactType("HK1000010T",
                  EppDomain.CONTACT_TYPE_BILLING));
      cmd.removeContact(new EppContactType("HK1000004T",
                  EppDomain.CONTACT_TYPE_BILLING));

hkExtension hk = new hkExtension();
hk.addExtension("bundledomain.com.hk",
hkExtension.BUNDLE_DOMAIN_NAME);
hk.addExtension( "Reseller Info", hkExtension.RESELLER_INFO );
cmd.setEppExtension(hk);

// sends epp command to server and get response
EppResponse res = EppChannel.send(cmd);

if( res != null ) {
      if( res.success() ) {
            System.out.println("Transfer Ownership Completed
Successfully");
      } else {
            printErrors(res.getResult());
      }
} else {
      // throw handle exception
}
```

3.2.3.7 Bundle Domain

This method expected that the domain object be populated with the unique identifier of the domain to be updated and the unique identifier of the domain to be bundled. The following list shows fields that are required for the method called.

Objects

EppCommandUpdateDomain cmd = (EppCommandUpdateDomain) EppCommand.update(EppObject.DOMAIN, domainname, channel.getClientId());
hkExtension hk = new hkExtension();

Mandatory	Description	Value	Parameter
Yes	Domain Name	mydomain.com.hk	EppCommandUpdateDomain cmd = (EppCommandUpdateDomain) EppCommand.update(EppObject.DOMAIN, %value, channel.getClientId());
Yes	Bundled Domain Name	mybundledomain.com.hk	hk.domainExtension(%value, hkExtension.NEW_BUNDLE_DOMAIN_NAME)

Post-Conditions

When completed successfully, a standard *EppResponse* object is returned.

Response Code

1001 ; command completed successfully; action pending
2001 ; command syntax error
2003 ; required parameter missing
2004 ; parameter value range error
2005 ; parameter value syntax error
2102 ; unimplemented option
2201 ; authorization error
2300 ; object pending transfer
2303 ; object does not exists
2304 ; object status prohibits operation
2306 ; parameter value policy error
2400 ; command failed

Sample Code

The following example shows the steps of performing domain bundling through the use of the domain client interface and command send method.

```
EppCommandUpdateDomain cmd = (EppCommandUpdateDomain)
EppCommand.update(EppObject.DOMAIN, "mydomain.com.hk",
channel.getClientId());

// Domain name to be bundled
hkExtension hk = new hkExtension();
hk.domainExtension("bundledomain.com.hk", hkExtension.
NEW_BUNDLE_DOMAIN_NAME);
cmd.setEppExtension(hk);

// sends epp command to server and get response
EppResponse res = EppChannel.send(cmd);

if( res != null ) {
      if( res.success() ) {
            System.out.println("Transfer Ownership Completed
Successfully");
      } else {
            printErrors(res.getResult());
      }
} else {
```

3.2.3.8 Info Domain

This method sends the EPP info domain command to retrieve domain name information.

Pre-Conditions

This method expects that the domain object be populated with the single domain name of the domain to be queried.

Objects

EppCommandInfo cmd = EppCommand.info(EppObject.DOMAIN, domainname, EppChannel.getClientId());

Mandatory	Description	Value	Parameter
Yes	Domain Name	mydomain.com.hk	EppCommandInfo cmd = EppCommand.info(EppObject.DOMAIN, %value, EppChannel.getClientId());

Post-Conditions

- When completed successfully, *EppResponseDataInfo, EppDomain* object is returned.

Response Code

1000 ; command completed successfully
2001 ; command syntax error
2003 ; required parameter missing
2004 ; parameter value range error
2005 ; parameter value syntax error
2201 ; authorization error
2303 ; object does not exists
2306 ; parameter value policy error
2400 ; command failed

Sample Code

The following example shows the steps of querying a domain through the use of the domain client interface and command send method.

```
String domainname="mydomain.com.hk";

EppCommandInfo cmd = EppCommand.info(EppObject.DOMAIN, domainname,
EppChannel.getClientId());

EppResponse res = EppChannel.send(cmd);

if( res != null ) {
      if( res.success() ) {
            EppResponseDataInfo res_data =
            (EppResponseDataInfo) res.getResponseData();

            if( res_data != null ) {
                  EppDomain Domain = (EppDomain) res_data.getObject();
                  //retrieve Domain attribute here
            } else {
                  // throw handle exception
            }
      } else {
          printErrors(res.getResult());
      }
} else {
      // throw handle exception
}
```

3.2.3.9 Transfer Domain

This method sends the EPP transfer domain command. There are 5 different operations that can be performed: query, request, approve, reject or cancel a transfer. A transfer is initiated by the registrar taking up the domain. Once a transfer has been requested, the current registrar on record will be notified by the system. The current registrar is then required to either approve or reject the transfer.

Pre-Conditions

This method requires that several attributes be set prior to execution. The following list shows fields that are required for the method.

Objects

EppCommandTransferDomain cmd = (EppCommandTransferDomain) EppCommand.transfer(EppObject.DOMAIN, domainname, EppChannel.getClientId());
EppAuthInfo Authinfo = new EppAuthInfo(EppAuthInfo.TYPE_PW, "123456");

Mandatory	Description	Value	Parameter
Yes	Domain Name	mydomain.com.hk	EppCommandTransferDomain cmd = (EppCommandTransferDomain) EppCommand.transfer(EppObject.DOMAIN, %value, EppChannel.getClientId());
Yes	Domain Authorisation Code	123456	EppAuthInfo Authinfo = new EppAuthInfo(EppAuthInfo.TYPE_PW, %value);
Yes	Operation	OPTYPE_REQUEST (refer operation legends below)	cmd.setOperation(%value);
Yes if bundle	Bundle Domain Name	Mybundledomain.com.hk	hk.domainExtension(%value, hkExtension.BUNDLE_DOMAIN_NAME);

Operation Legends

OPTYPE_REQUEST – Request for transfer
OPTYPE_APPROVE – Approve a transfer
OPTYPE_REJECT – Reject a Transfer
OPTYPE_CANCEL – Cancel a Transfer
OPTYPE_QUERY – Query a Transfer

Post-Conditions

- When completed successfully, an *EppResponseDataTransfer* object is returned.

Response Code:

1000 ; command completed successfully

2001 ; command syntax error
2003 ; required parameter missing
2004 ; parameter value range error
2005 ; parameter value syntax error
2102 ; unimplemented option
2104 ; billing failure
2106 ; object is not eligible for transfer
2201 ; authorization error
2202 ; invalid authorization information
2300 ; object pending transfer
2301 ; object not pending transfer
2303 ; object does not exists
2304 ; object status prohibits operation
2306 ; parameter value policy error
2400 ; command failed

Sample Code

The following example shows the steps of requesting domain transfer through the use of the domain client interface and command send method.

```
String operation="";
if (action.equals("query")) {
    operation = EppCommandTransfer.OPTYPE_QUERY;
} else if (action.equals("request")) {
    operation = EppCommandTransfer.OPTYPE_REQUEST;
} else if (action.equals("approve")) {
    operation = EppCommandTransfer.OPTYPE_APPROVE;
} else if (action.equals("reject")) {
    operation = EppCommandTransfer.OPTYPE_REJECT;
} else if (action.equals("cancel")) {
    operation = EppCommandTransfer.OPTYPE_CANCEL;
}

EppCommandTransferDomain cmd = (EppCommandTransferDomain)
      EppCommand.transfer(EppObject.DOMAIN, "mydomain.com.hk",
      EppChannel.getClientId());

EppAuthInfo Authinfo = new EppAuthInfo(EppAuthInfo.TYPE_PW,
"z2daTIy0");
      cmd.setAuthInfo(Authinfo);
      cmd.setOperation(operation);

hkExtension hk = new hkExtension();
      hk.domainExtension(mybundledomain.com.hk",
      hkExtension.BUNDLE_DOMAIN_NAME);

cmd.setEppExtension(hk);

EppResponse res = EppChannel.send(cmd);

if( res != null ) {
      if( res.success() ) {
            EppResponseDataTransfer res_data =
            (EppResponseDataTransfer) res.getResponseData();
            System.out.println("Transfer Request Completed
Successfully");
      } else {
```

3.2.3.10 Special Promotion for Cross Selling and Brand name protection

This method is used to register new domain at a discounted price based on the promotion requirement. The following list shows fields that are required for the method called.

Objects

EppCommandUpdateDomain cmd = (EppCommandUpdateDomain) EppCommand.update(EppObject.DOMAIN, domainname, channel.getClientId());
hkExtension hk = new hkExtension();

Mandatory	Description	Value	Parameter
Yes	Domain Name	mydomain.com.hk	EppCommandUpdateDomain cmd = (EppCommandUpdateDomain) EppCommand.update(EppObject.DOMAIN, %value, channel.getClientId());
Yes	Claimed Domain Name	Myd0main.hk	hk.domainExtension(%value, hkExtension.CROSS_SELLING_DOMAIN_NAME)

Post-Conditions

When completed successfully, a standard *EppResponse* object is returned.

Response Code

1000: command completed successfully
2001: command syntax error
2003: required parameter missing
2102: unimplemented option
2201: authorization error
2300: object pending transfer
2303: object does not exists
2304: object status prohibits operation
2306: parameter value policy error
2400: command failed
2850: System Error on Brand Protection Promotion

Sample Code

The following example shows the steps of registering the domain with the cross selling/brand name protection promotion through the use of the domain client interface and command send method.

```
EppCommandUpdateDomain cmd = (EppCommandUpdateDomain)
EppCommand.update(EppObject.DOMAIN, "mydomain.com.hk",
channel.getClientId());

// Domain name to be registered
hkExtension hk = new hkExtension();
hk.domainExtension("myd0main.hk", hkExtension.
CROSS_SELLING_DOMAIN_NAME);
cmd.setEppExtension(hk);

// sends epp command to server and get response
EppResponse res = EppChannel.send(cmd);

if( res != null ) {
      if( res.success() ) {
            System.out.println("Command completed successfully");
      } else {
            printErrors(res.getResult());
      }
} else {
      // throw handle exception
```

3.2.3.11 Update Domain DNSSEC

This method is used to add / remove DNSSEC record of a domain. The following list shows fields that are required for the method called.

Objects

EppCommandUpdateDomain cmd = (EppCommandUpdateDomain) EppCommand.update(EppObject.DOMAIN, domainname, channel.getClientId());
DnssecExtension dnssec = new DnssecExtension();
dnssec. addDnnsecDataToAdd(keyTag, alg, digestType, digest);
dnssec.addDnnsecDataToRemove(keyTag, alg, digestType, digest);
cmd.setEppExtension(dnssec);

Mandatory	Description	Value	Parameter
Yes	Domain Name	mydomain.com.hk	EppCommandUpdateDomain cmd = (EppCommandUpdateDomain) EppCommand.update(EppObject.DOMAIN, %value, channel.getClientId());
Yes if no DNSSEC Record to remove	DNSSEC Record to add (keytag, algorithm, digest type, digest)	12345, 5, 1, 38EC35D5B3A34B44C39B38EC35D5B3A34B44C39C	dnssec.addDnnsecDataToAdd(keyTag, alg, digestType, digest);
Yes if no DNSSEC Record to add	DNSSEC Record to remove (keytag, algorithm, digest type, digest)	12345, 5, 1, 38EC35D5B3A34B44C39B38EC35D5B3A34B44C39C	dnssec.addDnnsecDataToRemove(keyTag, alg, digestType, digest);

Post-Conditions

When completed successfully, a standard *EppResponse* object is returned.

Response Code

1000: command completed successfully
1001: command completed successfully; action pending
2001: command syntax error
2003: required parameter missing
2004: parameter value range error
2005: parameter value syntax error
2102: unimplemented option
2201: authorization error
2300: object pending transfer
2303: object does not exists
2304: object status prohibits operation
2306: parameter value policy error
2400: command failed

2700: System Error on DNSSEC

Sample Code

The following example shows the steps of adding / removing DNSSEC record of a domain through the use of the domain client interface and command send method.

```
EppCommandUpdateDomain cmd = (EppCommandUpdateDomain)
EppCommand.update(EppObject.DOMAIN, "mydomain.com.hk",
channel.getClientId());

// DNNSEC record to be added / removed
DnssecExtension dnssec = new DnssecExtension();
dnssec. addDnnsecDataToAdd(keyTag, alg, digestType, digest);
dnssec.addDnnsecDataToRemove(keyTag, alg, digestType, digest);
cmd.setEppExtension(dnssec);

// sends epp command to server and get response
EppResponse res = EppChannel.send(cmd);

if( res != null ) {
      if( res.success() ) {
            System.out.println("Command completed successfully");
      } else {
            printErrors(res.getResult());
      }
} else {
      // throw handle exception
```

3.2.4 Poll Interface

This interface is used to send a poll command to either request a poll message or send a poll message acknowledgement. Once the acknowledgement is sent, the current active record in the queue will be cleared and registrar will be able to retrieve the next active record in the queue. Without acknowledgement, the same record will be returned from the queue when the request command is sent.

3.2.4.1 Send Poll

This method sends the EPP send poll command to acquire domain transfer status.

Pre-Conditions

The following list shows fields that are required for the method.

Objects

EppCommandPoll cmd = EppCommand.poll(EppCommandPoll.OPTYPE_REQ, "", EppChannel.getClientId());

Mandatory	Description	Value	Parameter
Yes	Poll Command	EppCommandPoll.OPTYPE_REQ or EppCommandPoll.OPTYPE_ACK	EppCommandPoll cmd = EppCommand.poll(%value, "", EppChannel.getClientId());
Yes	Message ID	Poll message id	EppCommandPoll cmd = EppCommand.poll(EppCommandPoll.OPTYPE_ACK, %value, EppChannel.getClientId());

Post-Conditions

A poll message is contained in the EPPResponse when the poll operation is EPPSession.OP_REQ and the poll message removed from operation is EPPSession.OP_ACK

Response Code:

1000 ; command completed successfully
1300 ; command completed successfully; no messages
1301 ; command completed successfully; ack to dequeue
2001 ; command syntax error
2003 ; required parameter missing
2303 ; object does not exists
2308 ; data management policy violation
2400 ; command failed

Sample Code

The following example shows the steps of poll request and poll acknowledge command send method.

```
EppCommandPoll cmd = EppCommand.poll(EppCommandPoll.OPTYPE_REQ, "",
EppChannel.getClientId());

EppResponse res = EppChannel.send(cmd);

if( res != null ) {
      if( res.success() ) {
                EppResponseDataTransferDomain res_data =
(EppResponseDataTransferDomain) res.getResponseData();
                if(res_data!=null){
                    this.domName=res_data.getName();
                    this.trStatus=res_data.getTransferStatus();
                    this.reID=res_data.getRequestClientId();
                    this.reDate=res_data.getRequestDate();
                    this.acID=res_data.getActionClientId();
                    this.acDate=res_data.getActionDate();
                }
                this.qCount=res.getMessageQueued();
                this.qID=res.getMessageId();
                this.qDate=res.getMessageQueueUpdated();
      } else {
            printErrors(res.getResult());
      }
} else {
// throw handle exception
}
```

```
EppCommandPoll cmd = EppCommand.poll(EppCommandPoll.OPTYPE_ACK,
messageId, EppChannel.getClientId());

EppResponse res = EppChannel.send(cmd);

if( res != null ) {
      if( res.success() ) {
                this.qCount=res.getMessageQueued();
                this.qID=res.getMessageId();
                this.qDate=res.getMessageQueueUpdated();
      } else {
            printErrors(res.getResult());
      }
} else {
// throw handle exception
}
```

3.2.5 Tracking Interface

This interface is used to sends an info command to retrieve the latest statuses and updated details of a transaction that pending for action.

3.2.5.1 Query Tracking

This method sends the EPP info tracking command to retrieve tracking information.

Pre-Conditions

This method expects that the tracking object be populated with the tracking details of the tracking to be queried.

Objects

EppCommandInfo cmd = EppCommand.info(EppObject.TRACKING, trkNumber, EppChannel.getClientId());

Mandatory	Description	Value	Parameter
Yes	Tracking	NDN435632	EppCommandInfo cmd = EppCommand.info(EppObject.TRACKING, %value, EppChannel.getClientId());

Post-Conditions

- On success, *EppResponseDataInfo, EppTracking* object is returned, with the following attributes:
    - Tracking Number
    - Domain Name
    - Bundle Domain Name
    - Tracking Status
    - Document Approval Status
    - Payment Status
    - Apply Date
    - Transaction Activation or Rejected Date
    - Is Warning[1]
    - Is Conflicker[2]

Response Code

1000 ; command completed successfully
2001 ; command syntax error
2003 ; required parameter missing

[1] For more information about domain warning pattern, please refer to HKIRC Registration Policies (Section 7.4 to 7.8). URL: https://www.hkirc.hk/content.jsp?id=33

[2] For more information about Conflicker, please refer to Hong Kong Computer Emergency Response Team ("HKCERT") web site. URL: https://www.hkcert.org/my_url/en/articles/09032501

2004 ; parameter value range error
2005 ; parameter value syntax error
2201 ; authorization error
2303 ; object does not exists
2306 ; parameter value policy error
2400 ; command failed

Sample Code

The following example shows the steps of querying a tracking through the use of the tracking client interface and command send method.

```
String trkNumber = ”NDN435632”;

EppCommandInfo cmd = EppCommand.info(EppObject.TRACKING, trkNumber,
EppChannel.getClientId());

EppResponse res = EppChannel.send(cmd);

if( res != null ) {
      if( res.success() ) {
            EppResponseDataInfo res_data =
            (EppResponseDataInfo) res.getResponseData();

            if( res_data != null ) {
                  EppTracking obj = (EppTracking) res_data.getObject();
                  //retrieve tracking attribute here
            } else {
                  // throw handle exception
            }
      } else {
          printErrors(res.getResult());
      }
} else {
      // throw handle exception
}
```

3.2.5.2 Create RAC Tracking

This method sends the EPP create RAC tracking command to initiate an authorization code request of the specified domain name.

Pre-Conditions

The following list shows the required field for the method.

Objects

EppCommandCreate cmd = EppCommand.createRAC(EppObject.DOMAINNAME, channel.getClientId());

Mandatory	Description	Value	Parameter
Yes	Domain Name	EXAMPLE.COM.HK	*EppCommandCreate* cmd = *EppCommand.createRAC* (%VALUE, EppChannel.getClientId());

Post-Conditions

- On success, *EppResponseDataRAC, EppTracking* object is returned, with the following attribute:
    - Tracking Number

Response Code

1000 ; command completed successfully
2003 ; required parameter missing
2201 ; authorization error
2303 ; object does not exists
2400 ; command failed

Sample Code

The following example shows the steps of creating a RAC tracking through the use of the tracking client interface and command send method.

```
String domainName = ”EXAMPLE.COM.HK”;

EppCommandCreate cmd = EppCommand.createRAC(domainName,
EppChannel.getClientId());

EppResponse res = EppChannel.send(cmd);

if( res != null ) {
      if( res.success() ) {
            EppResponseDataRAC res_data = (EppResponseDataRAC)
      res.getResponseData();

            if( res_data != null ) {
                  EppTracking obj = (EppTracking) res_data.getObject();
                  //retrieve tracking attribute here
            } else {
                  // throw handle exception
            }
      } else {
          printErrors(res.getResult());
      }
} else {
      // throw handle exception
}
```

3.2.5.3 Update Tracking Document Status

This method sends the EPP update tracking document status command to initiate a tracking document status update request.

Pre-Conditions

The following list shows the required field for the method.

Objects

EppCommandUpdateDocstatus cmd = EppCommand.update(EppObject.TRACKING, trkNumber, channel.getClientId());

Mandatory	Description	Value	Parameter
Yes	Tracking Number	NDN123456	*EppCommandUpdateDocstatus cmd = EppCommand.update(EppObject.TRACKING, %VALUE, channel.getClientId());*
Yes	Document Status	A, R, P	*cmd.setDocStatus(%VALUE);*

Domain Status

A = Approve
R = Reject
P = Pending

Post-Conditions

When completed successfully, a standard *EppResponse* object is returned.

Response Code

1000 ; command completed successfully
2001 ; command syntax error
2003 ; required parameter missing
2004 ; parameter value range error
2005 ; parameter value syntax error
2102 ; unimplemented option
2201 ; authorization error
2303 ; object does not exists
2304 ; object status prohibits operation
2400 ; command failed

Sample Code

The following example shows the steps of updating a tracking document status through the use of the tracking client interface and command send method.

```
String trkNumber = ”NDN123456”;
String docStatus = ”A”;

EppCommandUpdateDocstatus cmd =
(EppCommandUpdateDocstatus)EppCommand.update(EppObject.TRACKING,
trkNumber, channel.getClientId());
cmd.setDocStatus(docStatus);

EppResponse res = EppChannel.send(cmd);

if( res != null ) {
      if( res.success() )
      {
            this.resultdesc = "Command completed successfully";
            this.resultcode = "1000";
      }
      else
      {
            sessionObj.printErrors(res.getResult());
            this.resultdesc = sessionObj.getresultdesc();
            this.resultcode = sessionObj.getresultcode();
      }
} else {
      // throw handle exception
`
```

Appendix

3.3 Country Code

AF:AFGHANISTAN
AL:ALBANIA
DZ:ALGERIA
AS:AMERICAN SAMOA
AD:ANDORRA
AO:ANGOLA
AI:ANGUILLA
AQ:ANTARCTICA
AG:ANTIGUA AND BARBUDA
AR:ARGENTINA
AM:ARMENIA
AW:ARUBA
AU:AUSTRALIA
AT:AUSTRIA
AZ:AZERBAIDJAN
BS:BAHAMAS
BH:BAHRAIN
BD:BANGLADESH
BB:BARBADOS
BY:BELARUS
BE:BELGIUM
BZ:BELIZE
BJ:BENIN
BM:BERMUDA
BT:BHUTAN
BO:BOLIVIA
BA:BOSNIA-HERZEGOVINA
BW:BOTSWANA
BV:BOUVET ISLAND
BR:BRAZIL
IO:BRITISH INDIAN OCEAN
BN:BRUNEI DARUSSALAM
BG:BULGARIA
BF:BURKINA FASO
BI:BURUNDI
KH:CAMBODIA
CM:CAMEROON
CA:CANADA
CV:CAPE VERDE
KY:CAYMAN ISLANDS
CF:CENTRAL AFRICAN REPUB
TD:CHAD
CL:CHILE
CN:CHINA
CX:CHRISTMAS ISLAND
CC:COCOS (KEELING) ISLANDS
CO:COLOMBIA
KM:COMOROS
CG:CONGO
CK:COOK ISLANDS
CR:COSTA RICA
HR:CROATIA
CU:CUBA
CY:CYPRUS
CZ:CZECH REPUBLIC
DK:DENMARK
DJ:DJIBOUTI
DM:DOMINICA
DO:DOMINICAN REPUBLIC
TP:EAST TIMOR
EC:ECUADOR
EG:EGYPT
SV:EL SALVADOR
GQ:EQUATORIAL GUINEA
EE:ESTONIA
ET:ETHIOPIA
FK:FALKLAND ISLANDS
FO:FAROE ISLANDS
FJ:FIJI
FI:FINLAND
CS:FORMER CZECHOSLOVAKIA
SU:FORMER USSR
FR:FRANCE
FX:FRANCE (EUROPEAN TERRITORIES)
GF:FRENCH GUYANA
TF:FRENCH SOUTHERN TERRITORIES
GA:GABON
GM:GAMBIA
GE:GEORGIA
DE:GERMANY
GH:GHANA
GI:GIBRALTAR
GB:GREAT BRITAIN
GR:GREECE
GL:GREENLAND
GD:GRENADA
GP:GUADELOUPE (FRENCH)
GU:GUAM (USA)
GT:GUATEMALA
GW:GUINEA BISSAU
GY:GUYANA
HT:HAITI
HM:HEARD AND MCDONALD ISLANDS
HN:HONDURAS
HK:HONG KONG
HU:HUNGARY
IS:ICELAND
IN:INDIA
ID:INDONESIA
IR:IRAN
IQ:IRAQ
IE:IRELAND
IL:ISRAEL
IT:ITALY
CI:IVORY COAST (COTE D'I)
JM:JAMAICA
JP:JAPAN
JO:JORDAN
JF:JOTHAN FRAKES ISLANDS
KZ:KAZAKHSTAN
KE:KENYA
KI:KIRIBATI
KW:KUWAIT
KG:KYRGYZSTAN

LA:LAOS
LV:LATVIA
LB:LEBANON
LS:LESOTHO
LR:LIBERIA
LY:LIBYA
LI:LIECHTENSTEIN
LT:LITHUANIA
LU:LUXEMBOURG
MO:MACAU
MK:MACEDONIA
MG:MADAGASCAR
MW:MALAWI
MY:MALAYSIA
MV:MALDIVES
ML:MALI
MT:MALTA
MH:MARSHALL ISLANDS
MQ:MARTINIQUE (FRENCH)
MR:MAURITANIA
MU:MAURITIUS
YT:MAYOTTE
MX:MEXICO
FM:MICRONESIA
MD:MOLDAVIA
MC:MONACO
MN:MONGOLIA
MS:MONTSERRAT
MA:MOROCCO
MZ:MOZAMBIQUE
MM:MYANMAR
NA:NAMIBIA
NR:NAURU
NP:NEPAL
NL:NETHERLANDS
AN:NETHERLANDS ANTILLES
NC:NEW CALEDONIA (FRENCH)
NZ:NEW ZEALAND
NI:NICARAGUA
NE:NIGER
NG:NIGERIA
NU:NIUE
NF:NORFOLK ISLAND
KP:NORTH KOREA
MP:NORTHERN MARIANA ISLANDS
NO:NORWAY
OM:OMAN
PK:PAKISTAN
PW:PALAU
PA:PANAMA
PG:PAPUA NEW GUINEA
PY:PARAGUAY
PE:PERU
PH:PHILIPPINES
PN:PITCAIRN ISLAND
PL:POLAND
PF:POLYNESIA (FRENCH)
PT:PORTUGAL
ZN:PRINCE NIZAM ZAMBRI ISLE
PR:PUERTO RICO
QA:QATAR
RE:REUNION (FRENCH)
RO:ROMANIA
RU:RUSSIAN FEDERATION
RW:RWANDA
GS:S. GEORGIA & S. SANDWICH ISLANDS
SH:SAINT HELENA
KN:SAINT KITTS & NEVIS
LC:SAINT LUCIA
PM:SAINT PIERRE AND MIQU
ST:SAINT TOME (SAO TOME)
VC:SAINT VINCENT & GRENA
WS:SAMOA
SM:SAN MARINO
SA:SAUDI ARABIA
SN:SENEGAL
SC:SEYCHELLES
SL:SIERRA LEONE
SG:SINGAPORE
SK:SLOVAK REPUBLIC
SI:SLOVENIA
SB:SOLOMON ISLANDS
SO:SOMALIA
ZA:SOUTH AFRICA
KR:SOUTH KOREA
ES:SPAIN
LK:SRI LANKA
SD:SUDAN
SR:SURINAME
SJ:SVALBARD AND JAN MAYE
SZ:SWAZILAND
SE:SWEDEN
CH:SWITZERLAND
SY:SYRIA
TJ:TADJIKISTAN
TW:TAIWAN
TZ:TANZANIA
TH:THAILAND
TG:TOGO
TK:TOKELAU
TO:TONGA
TT:TRINIDAD AND TOBAGO
TN:TUNISIA
TR:TURKEY
TM:TURKMENISTAN
TC:TURKS AND CAICOS ISLANDS
TV:TUVALU
UG:UGANDA
UA:UKRAINE
AE:UNITED ARAB EMIRATES
UK:UNITED KINGDOM
US:UNITED STATES
UY:URUGUAY
UM:USA MINOR OUTLYING ISANDS
UZ:UZBEKISTAN
VU:VANUATU
VA:VATICAN CITY STATE
VE:VENEZUELA
VN:VIETNAM
VG:VIRGIN ISLANDS (BRITISH)
VI:VIRGIN ISLANDS (USA)
WF:WALLIS AND FUTUNA ISLANDS
EH:WESTERN SAHARA
YE:YEMEN

YU:YUGOSLAVIA
ZR:ZAIRE
ZM:ZAMBIA
ZW:ZIMBABWE

3.4 Others

1. Individual Registrant Document Type

HKID:Hong Kong Identity Number
OTHID:Other's Country Identity Number
PASSNO:Passport No.
BIRTHCERT:Birth Certificate
OTHIDV:Others Individual Document

2. Organization Registrant Document Type

BR:Business Registration Certificate
CI:Certificate of Incorporation
CRS:Certificate of Registration of a School
HKSARG:Hong Kong Special Administrative Region Government Department
HKORDINANCE:Ordinance of Hong Kong
OTHORG:Others Organization Document

3. Registrant Industry Type

0:None (Let the value of the parameter '0' if choosing 'None' for registrant industry type)
010100:Plastics, Petro-Chemicals, Chemicals - Plastics & Plastic Products
010200:Plastics, Petro-Chemicals, Chemicals - Rubber & Rubber Products
010300:Plastics, Petro-Chemicals, Chemicals - Fibre Materials & Products
010400:Plastics, Petro-Chemicals, Chemicals - Petroleum, Coal & Other Fuels
010500:Plastics, Petro-Chemicals, Chemicals - Chemicals & Chemical Products
020100:Metals, Machinery, Equipment - Metal Materials & Treatment
020200:Metals, Machinery, Equipment - Metal Products
020300:Metals, Machinery, Equipment - Industrial Machinery & Supplies
020400:Metals, Machinery, Equipment - Precision & Optical Equipment
020500:Metals, Machinery, Equipment - Moulds & Dies
030100:Printing, Paper, Publishing - Printing, Photocopying, Publishing
030200:Printing, Paper, Publishing - Paper, Paper Products
040100:Construction, Decoration, Environmental Engineering - Construction Contractors
040200:Construction, Decoration, Environmental Engineering - Construction Materials
040300:Construction, Decoration, Environmental Engineering - Decoration Materials
040400:Construction, Decoration, Environmental Engineering - Construction, Safety Equipment & Supplies
040500:Construction, Decoration, Environmental Engineering - Decoration, Locksmiths, Plumbing & Electrical Works
040600:Construction, Decoration, Environmental Engineering - Fire Protection Equipment & Services
040700:Construction, Decoration, Environmental Engineering - Environmental Engineering, Waste Reduction
050100:Textiles, Clothing & Accessories - Textiles, Fabric
050200:Textiles, Clothing & Accessories - Clothing
050300:Textiles, Clothing & Accessories - Uniforms, Special Clothing
050400:Textiles, Clothing & Accessories - Clothing Manufacturing Accessories
050500:Textiles, Clothing & Accessories - Clothing Processing & Equipment
050600:Textiles, Clothing & Accessories - Fur, Leather & Leather Goods
050700:Textiles, Clothing & Accessories - Handbags, Footwear, Optical Goods, Personal Accessories
060100:Electronics, Electrical Appliances - Electronic Equipment & Supplies
060200:Electronics, Electrical Appliances - Electronic Parts & Components
060300:Electronics, Electrical Appliances - Electrical Appliances, Audio-Visual Equipment
070100:Houseware, Watches, Clocks, Jewellery, Toys, Gifts - Kitchenware, Tableware
070200:Houseware, Watches, Clocks, Jewellery, Toys, Gifts - Bedding
070300:Houseware, Watches, Clocks, Jewellery, Toys, Gifts - Bathroom, Cleaning Accessories
070400:Houseware, Watches, Clocks, Jewellery, Toys, Gifts - Household Goods
070500:Houseware, Watches, Clocks, Jewellery, Toys, Gifts - Wooden, Bamboo & Rattan Goods
070600:Houseware, Watches, Clocks, Jewellery, Toys, Gifts - Home Furnishings, Arts & Crafts
070700:Houseware, Watches, Clocks, Jewellery, Toys, Gifts - Watches, Clocks
070800:Houseware, Watches, Clocks, Jewellery, Toys, Gifts - Jewellery Accessories
070900:Houseware, Watches, Clocks, Jewellery, Toys, Gifts - Toys, Games, Gifts
080100:Business & Professional Services, Finance - Accounting, Legal Services
080200:Business & Professional Services, Finance - Advertising, Promotion Services
080300:Business & Professional Services, Finance - Consultancy Services

080400:Business & Professional Services, Finance - Translation, Design Services
080500:Business & Professional Services, Finance - Cleaning, Pest Control Services
080600:Business & Professional Services, Finance - Security Services
080700:Business & Professional Services, Finance - Trading, Business Services
080800:Business & Professional Services, Finance - Employment Services
080900:Business & Professional Services, Finance - Banking, Finance, Investment
081000:Business & Professional Services, Finance - Insurance
081100:Business & Professional Services, Finance - Property, Real Estate
090100:Transportation, Logistics - Land Transport, Motorcars
090200:Transportation, Logistics - Sea Transport, Boats
090300:Transportation, Logistics - Air Transport
090400:Transportation, Logistics - Moving, Warehousing, Courier & Logistics Services
090500:Transportation, Logistics - Freight Forwarding
100100:Office Equipment, Furniture, Stationery, Information Technology - Office, Commercial Equipment & Supplies
100200:Office Equipment, Furniture, Stationery, Information Technology - Office & Home Furniture
100300:Office Equipment, Furniture, Stationery, Information Technology - Stationery & Educational Supplies
100400:Office Equipment, Furniture, Stationery, Information Technology - Telecommunication Equipment & Services
100500:Office Equipment, Furniture, Stationery, Information Technology - Computers, Information Technology
110100:Food, Flowers, Fishing & Agriculture - Food Products & Supplies
110200:Food, Flowers, Fishing & Agriculture - Beverages, Tobacco
110300:Food, Flowers, Fishing & Agriculture - Restaurant Equipment & Supplies
110400:Food, Flowers, Fishing & Agriculture - Flowers, Artificial Flowers, Plants
110500:Food, Flowers, Fishing & Agriculture - Fishing
110600:Food, Flowers, Fishing & Agriculture - Agriculture
120100:Medical Services, Beauty, Social Services - Medicine & Herbal Products
120200:Medical Services, Beauty, Social Services - Medical & Therapeutic Services
120300:Medical Services, Beauty, Social Services - Medical Equipment & Supplies
120400:Medical Services, Beauty, Social Services - Beauty, Health
120500:Medical Services, Beauty, Social Services - Personal Services
120600:Medical Services, Beauty, Social Services - Organizations, Associations
120700:Medical Services, Beauty, Social Services - Information, Media
120800:Medical Services, Beauty, Social Services - Public Utilities
120900:Medical Services, Beauty, Social Services - Religion, Astrology, Funeral Services
130100:Culture, Education - Music, Arts
130200:Culture, Education - Learning Instruction & Training
130300:Culture, Education - Elementary Education
130400:Culture, Education - Tertiary Education, Other Education Services
130500:Culture, Education - Sporting Goods
130600:Culture, Education - Sporting, Recreational Facilities & Venues
130700:Culture, Education - Hobbies, Recreational Activities
130800:Culture, Education - Pets, Pets Services & Supplies
140101:Dining, Entertainment, Shopping, Travel - Restaurant Guide - Chinese
140102:Dining, Entertainment, Shopping, Travel - Restaurant Guide - Asian
140103:Dining, Entertainment, Shopping, Travel - Restaurant Guide - Western
140200:Dining, Entertainment, Shopping, Travel - Catering Services, Eateries
140300:Dining, Entertainment, Shopping, Travel - Entertainment Venues
140400:Dining, Entertainment, Shopping, Travel - Entertainment Production & Services
140500:Dining, Entertainment, Shopping, Travel - Entertainment Equipment & Facilities
140600:Dining, Entertainment, Shopping, Travel - Shopping Venues
140700:Dining, Entertainment, Shopping, Travel - Travel, Hotels & Accommodation

4. Contact Type

1:Registrant
2:Administrative
3:Technical
4:Billing

5. Transfer Operation

request:Gaining Registrar requests for a domain transfer transaction
query:query for a domain whether in a queue of transfer

approve:Loosing Registrar approves the request of a domain transfer
reject:Loosing Registrar rejects the request of a domain transfer
cancel:Gaining Reseller cancels a domain transfer transaction