Space Details:

Key	CSJH
Name	Cryptshare Java API Handbuch
Description	
Created by	hartwigr (Jun 13, 2016)

Available Pages:

Cryptshare Java API Handbuch : Voraussetzungen

Created by René Hartwig, last modified on May 23, 2018

Gültige Lizenz

Um die Cryptshare Java API mit Ihrem Cryptshare Server nutzen zu können, benötigen Sie eine gültige Lizenz, sowohl für den Server selbst, als auch für die Java API. Diese Lizenz muss auf dem Cryptshare Server installiert sein. Den Status Ihrer Lizenz können Sie im Status-Bereich des Cryptshare Servers einsehen:

Product Name	Status
Cryptshare Core Application	licensed
Cryptshare for Outlook	licensed
Cryptshare for Notes	licensed
Cryptshare Robot	licensed
Java API	licensed
.NET API	licensed

WSDL Zugriff

Die Cryptshare Java API verwendet einen **WSDL Webservice** um die erforderlichen Operationen durchführen zu können. Dafür ist der Zugriff auf den Server über **Port 80 und Port 443** erforderlich. Der Server stellt zwei Webservices bereit, die für den erfolgreich Zugriff benötigt werden. Einer zur Behandlung allgemeiner Service-Requests und ein zweiter für den Transfer der Daten:

- https://<server_url>/service/**serviceV2?wsdl**
- https://<server_url>/service/**transferV2?wsdl**

Kann eines der beiden Interfaces nicht erreicht werden, so ist eine Nutzung der Java API nicht möglich.

Java Laufzeitumgebung

Für die Nutzung der Java API ist eine **Java Laufzeitumgebung v1.6 oder höher** erforderlich.

Verwandte Knowledgebase Artikel

- KeyNotFoundException after confirming the transfer dialog of a Cryptshare transfer
- Error message "System Error. Code: 18." when performing Cryptshare transfer
- Error message "The path is not of a legal form." when performing Cryptshare transfer
- Performance issues with Outlook when using Cryptshare for Office 365 & Outlook
- Cryptshare for Outlook does not recognize the incoming verification e-mail
- Exception with message "LoadServerSettingsOnStartupAsync" is thrown when launching Outlook
-

- Office application crashes shortly after launching
- Outlook crashes when switching to a certain folder
- Error message "MAPI_E_INVALID_PARAMETER" when user performs a transfer
- Error message "Some components of Cryptshare for Outlook V2 don't support [...]" when launching Outlook with the Add-in
- Several "Get List Of All User Account Exception" errors in the log file
- AccessViolationException when adding attachments via drag and drop
- Shared mailbox is not recognized by the Add-in
- User receives "System.MissingMethodException" exception when launching an Office application
- The Cryptshare Server becomes unresponsive under high load.

Befine Solutions AG - Schwarzwaldstr. 151 - 79102 Freiburg - GERMANY
Technical Support: Phone +49 761 389 13 100 - E-Mail: support@cryptshare.com

Cryptshare Java API Handbuch : Verifizierung

Created by René Hartwig, last modified by Simon Erhardt on Aug 03, 2018

Verifizierungsverfahren

Der größte Teil an Diensten, welche vom Cryptshare Server angeboten werden, erfordert eine Form der Verifizierung bevor der Dienst in Anspruch genommen werden kann. Dabei kann zwischen zwei unterschiedlichen Verfahren gewählt werden: Absender-Verifizierung und Client-Verifizierung.

- Ist der Cryptshare-Server so eingestellt, dass die Absender-Verifizierung erforderlich ist, so muss jede E-Mail Adresse, die als Absender verwendet wird, separat verifiziert werden.
- Mit der Client-Verifizierung wird der komplette Host, auf welchem sich die Client-Anwendung befindet, verifiziert. In diesem Falle können die Dienste des Cryptshare Servers unter Verwendung beliebiger Absenderadressen in Anspruch genommen werden ohne zusätzliche Verifizierungen durchführen zu müssen.

Das verwendete Verifizierungsverfahren kann im Bereich 'Zusatzprodukte' in der Administrationsoberfläche des Cryptshare Servers eingestellt werden:

Prüfen der Verifizierung

Welches Verifizierungsverfahren derzeitig eingestellt ist kann mit API-Methoden geprüft werden. Außerdem kann überprüft werden ob bestimmte Absenderadressen, oder im Falle der Client-Verifizierung, der Host selbst verifiziert ist.
Wie bei allen API-Operationen muss zuerst eine Client-Instanz angelegt werden, wie unter API-Funktionen beschrieben.
Anschließend kann mittels **Client#checkVerification()** der Verifizierungszustand überprüft werden. Dabei wird ein '**VerificationStatus**'-Objekt zurückgeliefert, welches den angefragten Zustand enthält.
Liefert **VerificationStatus#isSenderVerification()** 'true' so ist das Verifizierungsverfahren am Server auf 'Absender-Verifizierung' eingestellt, andernfalls ist der Server auf die Verwendung der Client-Verifizierung eingestellt.
Mit **#isVerified()** kann überprüft werden, ob die aktuelle Absenderadresse verifiziert ist, beziehungsweise, im Falle der Client-Verifizierung, ob der verwendete Host verifiziert ist.

Beispiel: Überprüfen der Verifizierung

```
// Schritt 1: Erzeugen einer Client-Instanz

// Anlegen der URL zu Ihrem Cryptshare Server
WebServiceUri serviceUri = new WebServiceUri("https://cryptshare.server.com");

// Erzeugen der Verbindung zum Cryptshare Server
CryptshareConnection connection = new CryptshareConnection(serviceUri);

// Erzeugen der Client-Instanz unter Verwendung der Absenderadresse,
// der Verbindung zum Server und des Pfades für denn lokalen Verifizierungsspeicher.
Client client = new Client("sender_email@server.com", connection,
```

```
"C:\\temp\\client.store");

// Schritt 2: Verwenden der Service-Methoden

// Nun kann der aktuellen Verifizierungszustand abgefragt werden
VerificationStatus result = client.checkVerification();

if (result.isSenderVerification()) {
    System.out.println("Verifizierungsverfahren ist 'Absenderverifizierung'.");
} else {
    System.out.println("Verifizierungsverfahren ist 'Client-Verifizierung'.");
}

if (result.isVerified()) {
    System.out.println("Dieser Absender ist verifiziert.");
} else {
    System.out.println("Dieser Absender ist NICHT verifiziert.");
}
```

Verwandte Knowledgebase-Artikel

- KeyNotFoundException after confirming the transfer dialog of a Cryptshare transfer
- Error message "System Error. Code: 18." when performing Cryptshare transfer
- Error message "The path is not of a legal form." when performing Cryptshare transfer
- Performance issues with Outlook when using Cryptshare for Office 365 & Outlook
- Cryptshare for Outlook does not recognize the incoming verification e-mail
- Exception with message "LoadServerSettingsOnStartupAsync" is thrown when launching Outlook
- Office application crashes shortly after launching
- Outlook crashes when switching to a certain folder
- Error message "MAPI_E_INVALID_PARAMETER" when user performs a transfer
- Error message "Some components of Cryptshare for Outlook V2 don't support [...]" when launching Outlook with the Add-in
- Several "Get List Of All User Account Exception" errors in the log file
- AccessViolationException when adding attachments via drag and drop
- Shared mailbox is not recognized by the Add-in
- User receives "System.MissingMethodException" exception when launching an Office application
- The Cryptshare Server becomes unresponsive under high load.

Befine Solutions AG - Schwarzwaldstr. 151 - 79102 Freiburg - GERMANY
Technical Support: Phone +49 761 389 13 100 - E-Mail: support@cryptshare.com

Cryptshare Java API Handbuch : Absender-Verifizierung

Created by René Hartwig, last modified on May 23, 2018

Bei der Absenderverifizierung muss für jede verwendete Absenderadresse eine Verifizierung durchgeführt werden. Dies stellt sicher, dass die verwendete Adresse, die zur Durchführung eines Transfers verwendet wird, gültig ist und zu dem jeweiligen Eigentümer gehört. Das ist wichtig da die Policyverwaltung des Cryptshare Servers die Berechtigungen und Einstellungen für einen Transfer von den verwendeten E-Mail Adressen abhängig macht.
Um eine Adresse zu verifizieren muss für diese Adresse ein Verifizierungs-Request an den Cryptshare Server geschickt werden. Der Cryptshare Server sendet dann eine E-Mail mit einem Verifizierungscode an die verwendete E-Mail Adresse. Anschließend kann dieser Code im lokalen Verifizierungsspeicher abgelegt werden. Ist der Verifizierungsvorgang abgeschlossen so wird für alle weiteren Requests, der gespeicherte Code an den Server übertragen und mit dem dortigen Datenbankeintrag abgeglichen. Dies stellt sicher, dass die verwendete E-Mail Adresse tatsächlich verifiziert ist.

Beispiel: Durchführen einer Absenderverifizierung

```
// Schritt 1: Erzeugen einer Client-Instanz

// Anlegen der URL zu Ihrem Cryptshare Server
WebServiceUri serviceUri = new WebServiceUri("https://cryptshare.server.com");

// Erzeugen der Verbindung zum Cryptshare Server
CryptshareConnection connection = new CryptshareConnection(serviceUri);

// Erzeugen der Client-Instanz unter Verwendung der Absenderadresse,
// der Verbindung zum Server und des Pfades für denn lokalen Verifizierungsspeicher.
Client client = new Client("sender_email@server.com", connection, "C:\temp");

// Schritt 2: Verwenden der Service-Methoden

// Nun kann der aktuellen Verifizierungszustand abgefragt werden
VerificationStatus result = client.checkVerification();

if (result.isSenderVerification()) {
    // Das Absender-Verifizierungsverfahren wird verwendet
    if (!result.isVerified()) {
        // Die verwendete E-Mail Adresse ist NICHT verifiziert,
        // daher wird ein neuer Verifizierungscode angefordert.
        client.requestVerification();

        // Der Cryptshare Server hat nun eine E-Mail mit dem Verifizierungscode
        // and die E-Mail Adresse "sender_email@server.com" geschickt.
        // Nun können Sie also beispielsweise den Benutzer dazu auffordern
        // den Code in der Kommandozeile einzugeben, oder bei Zugriff auf das
        // E-Mail Postfach kann automatisch nach einer E-Mail mit dem
        // Verifizierungscode gesucht werden.
        // Hier bitten wir nun den Benutzer den Code einzugeben:
        System.out.println("Bitte geben Sie den Verifizierungscode ein:");
        BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));
        String vericode = reader.readLine();
        reader.close();

        // Speichern des Verifizierungscodes im lokalen Verifizierungsspeicher
        client.storeVerification(vericode.trim());
    }
}
```

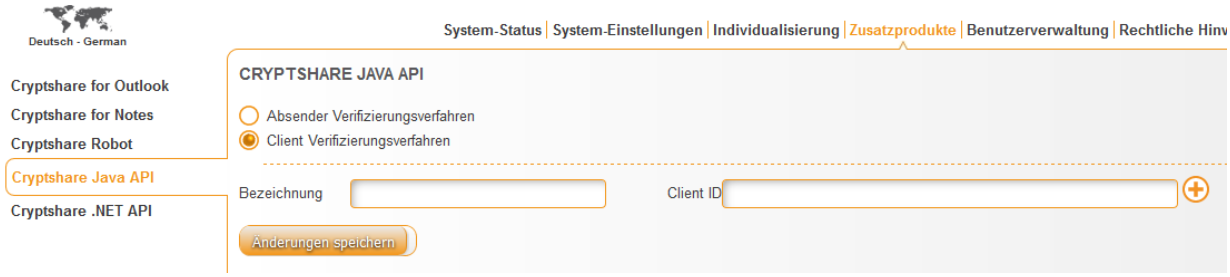
Verwandte Knowledegebase-Artikel

- KeyNotFoundException after confirming the transfer dialog of a Cryptshare transfer
- Error message "System Error. Code: 18." when performing Cryptshare transfer
- Error message "The path is not of a legal form." when performing Cryptshare transfer
- Performance issues with Outlook when using Cryptshare for Office 365 & Outlook
- Cryptshare for Outlook does not recognize the incoming verification e-mail
- Exception with message "LoadServerSettingsOnStartupAsync" is thrown when launching Outlook
- Office application crashes shortly after launching
- Outlook crashes when switching to a certain folder
- Error message "MAPI_E_INVALID_PARAMETER" when user performs a transfer
- Error message "Some components of Cryptshare for Outlook V2 don't support [...]" when launching Outlook with the Add-in
- Several "Get List Of All User Account Exception" errors in the log file
- AccessViolationException when adding attachments via drag and drop
- Shared mailbox is not recognized by the Add-in
- User receives "System.MissingMethodException" exception when launching an Office application
- The Cryptshare Server becomes unresponsive under high load.

Cryptshare Java API Handbuch : Client-Verifizierung

Created by René Hartwig, last modified on May 23, 2018

Bei Verwendung der Client-Verifizierung kann jede beliebige E-Mail Adresse für das Ausführen von Operationen verwendet werden, ohne dass der Absender den Verifizierungsvorgang, wie unter Absender-Verifizierung beschrieben, erneut durchführen muss.
In diesem Modus muss der Vorgang lediglich einmal für den verwendeten Host durchgeführt werden, anschließend können beliebige Absenderadressen die Funktionen der API nutzen.
Bitte beachten
Die Nutzung dieses Verifizierungsverfahrens **erhöht das Risiko des Missbrauchs**, da hierdurch jeder mit Zugriff auf den Host in der Lage ist die Funktionen der API zu nutzen, unabhängig davon welche Absenderadresse er verwendet. Es wird daher empfohlen dieses Verifizierungsverfahren nur dann zu verwenden wenn das Absender-Verifizierungsverfahren für Ihren Use-Case unzureichend ist.
Um Ihren Host zu verifizieren stellen Sie bitte zuerst sicher, dass am Server das Client-Verifizierungsverfahren eingestellt ist. Anschließend kann mit dem Methodenaufruf **Client#requestVerification()** eine neue Client-ID angefordert werden. Anschließend muss die erhaltene ID am Cryptshare Server eingetragen werden:



Geben Sie einen beschreibenden Text für den Client und die erhaltene Client-ID in die vorgesehenen Felder ein. Klicken Sie anschließend auf die Schaltfläche '+' um den neuen Eintrag anzulegen. Vergessen Sie bitte nicht die vorgenommenen Änderungen auch abzuspeichern.
Nun kann der verifizierte Host, ohne weitere Verifizierungsvorgänge für einzelne E-Mail Adressen, verwendet werden.

Beispiel: Durchführen einer Client-Verifizierung

```
 // Schritt 1: Erzeugen einer Client-Instanz

// Anlegen der URL zu Ihrem Cryptshare Server
WebServiceUri serviceUri = new WebServiceUri("https://cryptshare.server.com");

// Erzeugen der Verbindung zum Cryptshare Server
CryptshareConnection connection = new CryptshareConnection(serviceUri);

// Erzeugen der Client-Instanz unter Verwendung der Absenderadresse,
// der Verbindung zum Server und des Pfades für denn lokalen Verifizierungsspeicher.
Client client = new Client("sender_email@server.com", connection, "C:\temp");

// Schritt 2: Verwenden der Service-Methoden

// Nun kann der aktuellen Verifizierungszustand abgefragt werden
VerificationStatus result = client.checkVerification();

if (!result.isSenderVerification()) {
    // Das Client-Verifizierungsverfahren wird verwendet
    if (!result.isVerified()) {
        // Der Client ist NICHT verifiziert, also fordern wir eine neue Verifizierung an
        String clientID = client.requestVerification();

        // Wird #requestVerification() im Client-Verifizierungsmodus aufgerufen,
```

```
        // so wird vom Client automatisch eine neue Client-ID erzeugt und im
        // lokalen Verifizierungsspeicher abgelegt. Die erzeugte ID wird außerdem
        // in das verwendete Log geschrieben. Anschließend muss die erzeugte Client-ID
        // am Cryptshare Server registriert werden.
    }
}
```

Verwandte Knowledgebase-Artikel

- KeyNotFoundException after confirming the transfer dialog of a Cryptshare transfer
- Error message "System Error. Code: 18." when performing Cryptshare transfer
- Error message "The path is not of a legal form." when performing Cryptshare transfer
- Performance issues with Outlook when using Cryptshare for Office 365 & Outlook
- Cryptshare for Outlook does not recognize the incoming verification e-mail
- Exception with message "LoadServerSettingsOnStartupAsync" is thrown when launching Outlook
- Office application crashes shortly after launching
- Outlook crashes when switching to a certain folder
- Error message "MAPI_E_INVALID_PARAMETER" when user performs a transfer
- Error message "Some components of Cryptshare for Outlook V2 don't support [...]" when launching Outlook with the Add-in
- Several "Get List Of All User Account Exception" errors in the log file
- AccessViolationException when adding attachments via drag and drop
- Shared mailbox is not recognized by the Add-in
- User receives "System.MissingMethodException" exception when launching an Office application
- The Cryptshare Server becomes unresponsive under high load.

Befine Solutions AG - Schwarzwaldstr. 151 - 79102 Freiburg - GERMANY
Technical Support: Phone +49 761 389 13 100 - E-Mail: support@cryptshare.com

Cryptshare Java API Handbuch : Funktionen der API

Created by René Hartwig, last modified on May 23, 2018

Dieser Abschnitt beschreibt welche Methoden von der API angeboten werden und wie sie verwendet werden müssen um die gewünschten Operationen durchführen zu können.
Alle verfügbaren Servicemethoden werden von der **Client-Klasse** angeboten. Um also die Funktionen der API nutzen zu können, muss zuerst eine Instanz dieser Klasse erzeugt werden. Das Hauptanliegen der Java API ist der, einen möglichst einfachen Weg für die Übertragung von Cryptshare Transfers zu liefern. Daher erfordert die Erzeugung einer Client-Instanz immer die Angabe der Absenderadresse, die Verbindung zum Cryptshare Server in Form der CryptshareConnection-Klasse sowie die Verifizierungsinformationen.

Erzeugen einer Client-Instanz

Der erste Schritt für das Erzeugen einer Client-Instanz ist das Anlegen der **CryptshareConnection-Instanz** die mit der URL des Cryptshare Servers initialisiert wird. Der Konstruktor der **CryptshareConnection-Klasse** fordert die Angabe einer **WebServiceUri**, welche die URL des Cryptshare Servers erhält und damit die korrekten Service-URLs für den Zugriff auf den Cryptshare Server erzeugt. Sobald die CryptshareConnection-Instanz erzeugt ist kann Sie dem Konstruktor der Client-Klasse übergeben werden.
Neben der CryptshareConnection-Instanz fordert der Client-Konstruktor außerdem die Absenderadresse, sowie eine Pfadangabe zum Verifizierungsspeicher, oder alternativ den verschlüsselten Speicher in Form eines Byte-Arrays. Der Verifizierungsspeicher enthält die Verifizierungsinformationen des Clients, bzw. des Hosts, abhänig vom verwendeten Verifizierungsverfahren. Der Verifizierungsspeicher wird mit dem Client-Schlüssel verschlüsslet (Details unter Client-ID und Client-Schlüssel).
Für die Pfadangabe des Verifizierungsspeichers kann entweder der **vollständige Pfad mit Dateinamen**, ein **relativer Pfad mit Dateinamen** oder ein **Pfad ohne Dateinamen** angegeben werden. In letzterem Fall wird als Dateinamen 'client.store' verwendet. Existiert die Datei für den Verifiierungsspeicher nicht, so wird Sie automatisch von der Client-Instanz angelegt. Werden die Verifizierungsinformationen als Byte-Array übergeben, so erzeugt der Client keine Store-Datei. Die Daten werden dann lediglich mit dem Client-Schlüssel entschlüsselt und sind nur im Arbeitsspeicher hinterlegt.

Beispiele

Beispiel:Anlegen einer Client-Instanz mit Pfadangabe

```
// Erzeugen der WebServiceUri für den Cryptshare Server
WebServiceUri serviceUri = new WebServiceUri("https://cryptshare.server.com");

// Anlegen der CryptshareConnection für die WebServiceUri
CryptshareConnection connection = new CryptshareConnection(serviceUri);

// Anlegen der Client-Instanz mit der Absenderadresse, der
// CryptshareConnection, und dem Pfad zum Verifizierungsspeicher.
Client client = new Client("sender_email@server.com", connection, "C:\temp");
```

Verwandte Knowledgebase-Artikel

- KeyNotFoundException after confirming the transfer dialog of a Cryptshare transfer
- Error message "System Error. Code: 18." when performing Cryptshare transfer

- Error message "The path is not of a legal form." when performing Cryptshare transfer
- Performance issues with Outlook when using Cryptshare for Office 365 & Outlook
- Cryptshare for Outlook does not recognize the incoming verification e-mail
- Exception with message "LoadServerSettingsOnStartupAsync" is thrown when launching Outlook
- Office application crashes shortly after launching
- Outlook crashes when switching to a certain folder
- Error message "MAPI_E_INVALID_PARAMETER" when user performs a transfer
- Error message "Some components of Cryptshare for Outlook V2 don't support [...]" when launching Outlook with the Add-in
- Several "Get List Of All User Account Exception" errors in the log file
- AccessViolationException when adding attachments via drag and drop
- Shared mailbox is not recognized by the Add-in
- User receives "System.MissingMethodException" exception when launching an Office application
- The Cryptshare Server becomes unresponsive under high load.

Beispiel: Anlegen einer Client-Instanz mit Verifizierungsdaten als Byte-Array

```
 // Erzeugen der WebServiceUri für den Cryptshare Server
WebServiceUri serviceUri = new WebServiceUri("https://cryptshare.server.com");

// Anlegen der CryptshareConnection für die WebServiceUri
CryptshareConnection connection = new CryptshareConnection(serviceUri);

// Abrufen des Byte-Arrays mit den Verifizierungsdaten - die Methode
// #getStoreBytes() ist nicht Teil der Java API sondern dient lediglich
// als Platzhalter für einen vergleichbaren Vorgang für das Auslesen
// der Verifizierungsinformationen aus Ihrer Anwendung, beispielsweise
// das Auslesen aus einer Datenbank.
bytes[] storeBytes = getStoreBytes();

// Anlegen der Client-Instanz mit der Absenderadresse, der
// CryptshareConnection, und dem Pfad zum Verifizierungsspeicher.
Client client = new Client("sender_email@server.com", connection, storeBytes);
```

Sobad die Client-Instanz erzeugt wurde, können über die entsprechenden Objektmethoden die vorhandenen Serviceoperationen ausgeführt werden, wie in den enstprechenden Abschnitten dieses Handbuches beschrieben. Beachten Sie bitte, dass die meisten Operationen eine **gültige Verifizierung** erfordern. Wie eine Verifizierung durchzuführen ist, wird im Abschnitt 'Verifizierung' beschrieben.

Client-ID und Client-Schlüssel

Die Client-ID identifiziert einen verwendeten Host welcher die Java API nutzt. Die Client-ID enthält Daten die den Host eindeutig identifizieren, wie beispielsweise die MAC-Adresse. Die Client-ID wird bei der ersten Verwendung der API

angelegt und im lokalen Verifizierungsspeicher für die nachfolgende Nutzung abgelegt. Die Client-ID ist Voraussetzung für die Kommunikation mit dem Cryptshare Server.
Wird das Client-Verifizierungsverfahren verwendet, so wird diese Client-ID am Server registriert.
Der Client-Schlüssel wird, wie die Client-ID, auf Basis der verwendeten MAC-Adresse erzeugt und wird für die Verschlüsselung des Verifizierungsspeichers verwendet. Wenn also der Client den Verifizierungsspeicher öffnet, so erzeugt er den Client-Schlüssel und verwendet diesen um die Daten zu entschlüsseln.
Warnung
Da der Client-Schlüssel auf Basis der MAC-Adresse erzeugt wird, führt ein Austauschen der Netzwerkkarte zwangsläufig zu einem neuen Client-Schlüssel. Damit können die im Verifizierungsspeicher hinterlegten Daten nicht mehr entschlüsselt werden, da sie mit einem anderen Schlüssel erzeugt wurden. Vorhandene Verifizierungsinformationen gehen dadurch also verloren und es muss eine neue Verifizierung durchgeführt werden.
Um dies zu vermeiden, sollten Sie in Betracht ziehen die MAC-Adresse wieder auf die alte Adresse einzustellen, sofern dies möglich ist.

Cryptshare Java API Handbuch : Allgemeine Serverinformationen

Created by René Hartwig, last modified on May 23, 2018

Mit der Methode *client#requestServerData()* können allgemeine Servereinstellungen abgerufen werden. Die Methode gibt ein Objekt des Typs *ServerData* zurück welches folgende Daten enthält:

- Die offizielle URL des Cryptshare Servers
- Die Zeitzone in der sich der Server befindet
- Der noch verfügbare Festplattenplatz des Servers

Beispiel: Abrufen allgemeiner Serverinformationen

```
// 1) Client Instanz erzeugen
// WebServiceUri zum Cryptshare Server anlegen
WebServiceUri serviceUri = new WebServiceUri("https://cryptshare.server.com");

// CryptshareConnection-Instanz anlegen
CryptshareConnection connection = new CryptshareConnection(serviceUri);

// Client-Instanz mit Absenderadresse, CryptshareConnection und Verifizierungsspeicher
anlegen
Client client = new Client("sender_email@server.com", connection, "C:\temp");

// Serverdaten abrufen
ServerData serverData = client.requestServerData();

System.out.println("Offizielle Cryptshare Server URL: " + serverData.getServerUrl());
System.out.println("Server Zeitzone: " + serverData.getServerTimeZone());
System.out.println("Verfügbarer Festplattenplatz für neue Transfers (Uploadverzeichnis)
in bytes: "
                                        + serverData.getAvailableRetentionDiskSpace());

System.out.println("Verfügbarer Festplattenplatz für im Upload befindliche Transfers
(Temporäres Uploadverzeichnis) in bytes: "
                                        + serverData.getAvailableTempDiskSpace());
```

Verwandte Knowledgebaseartikel

- KeyNotFoundException after confirming the transfer dialog of a Cryptshare transfer
- Error message "System Error. Code: 18." when performing Cryptshare transfer
- Error message "The path is not of a legal form." when performing Cryptshare transfer
- Performance issues with Outlook when using Cryptshare for Office 365 & Outlook
- Cryptshare for Outlook does not recognize the incoming verification e-mail
- Exception with message "LoadServerSettingsOnStartupAsync" is thrown when launching Outlook
-

- Office application crashes shortly after launching
- Outlook crashes when switching to a certain folder
- Error message "MAPI_E_INVALID_PARAMETER" when user performs a transfer
- Error message "Some components of Cryptshare for Outlook V2 don't support [...]" when launching Outlook with the Add-in
- Several "Get List Of All User Account Exception" errors in the log file
- AccessViolationException when adding attachments via drag and drop
- Shared mailbox is not recognized by the Add-in
- User receives "System.MissingMethodException" exception when launching an Office application
- The Cryptshare Server becomes unresponsive under high load.

Befine Solutions AG - Schwarzwaldstr. 151 - 79102 Freiburg - GERMANY
Technical Support: Phone +49 761 389 13 100 - E-Mail: support@cryptshare.com

Cryptshare Java API Handbuch : Dateitransfer

Created by René Hartwig, last modified on May 23, 2018

Allgemeines

Transfers können synchron sowie asynchron über die API durchgeführt werden. Ein Transfer kann mehrere Dateien enthalten und wird anschließend auf den Cryptshare Server hochgeladen, verschlüsselt und den Empfängern zu Verfügung gestellt. Abhängig von den verwendeten Transfereinstellungen werden die Empfänger und der Absender nach der Bereitstellung des Transfers beanchrichtigt.
Bevor ein Transfer durchgeführt werden kann muss dieser, wie im Kapitel 'Transfer vorbereiten' beschrieben, vorbereitet werden. Anschließend kann der Transfer entweder synchron oder asynchron durchgeführt werden.

Standardservereinstellungen abrufen

Die Standardeinstellungen, wie beispielsweise die maximale Transfergröße, werden am Cryptshare Server festgelegt und gelten sofern sie nicht von spezifischen Policyeinstellungen übersteuert werden. Diese Einstellungen können mittels **#requestTransferData()** abgerufen werden. Das Rückgabe-Objekt enthält die folgenden Eisntellungen:

- Maximale Liegezeit für Transfers
- Maximale Transfergröße in MB

```
// Schritt 1: Erzeugen einer Client-Instanz

// Anlegen der URL zu Ihrem Cryptshare Server
WebServiceUri serviceUri = new WebServiceUri("https://cryptshare.server.com");

// Erzeugen der Verbindung zum Cryptshare Server
CryptshareConnection connection = new CryptshareConnection(serviceUri);

// Erzeugen der Client-Instanz unter Verwendung der Absenderadresse,
// der Verbindung zum Server und des Pfades für denn lokalen Verifizierungsspeicher.
Client client = new Client("sender_email@server.com", connection, "C:\temp");

// Allgemeine Transfereisntellungen abrufen
TransferSettings settings = client.requestTransferData();

System.out.println("Maximale Liegezeit in Tagen = " +
transferSettings.getStorageDuration());
System.out.println("Maximale Transfergröße in MB = " +
transferSettings.getTransferLimit());
```

Verwandte Knowledgebase-Artikel

- KeyNotFoundException after confirming the transfer dialog of a Cryptshare transfer
- Error message "System Error. Code: 18." when performing Cryptshare transfer
- Error message "The path is not of a legal form." when performing Cryptshare transfer
- Performance issues with Outlook when using Cryptshare for Office 365 & Outlook
- Cryptshare for Outlook does not recognize the incoming verification e-mail
-

Exception with message "LoadServerSettingsOnStartupAsync" is thrown when launching Outlook
- Office application crashes shortly after launching
- Outlook crashes when switching to a certain folder
- Error message "MAPI_E_INVALID_PARAMETER" when user performs a transfer
- Error message "Some components of Cryptshare for Outlook V2 don't support [...]" when launching Outlook with the Add-in
- Several "Get List Of All User Account Exception" errors in the log file
- AccessViolationException when adding attachments via drag and drop
- Shared mailbox is not recognized by the Add-in
- User receives "System.MissingMethodException" exception when launching an Office application
- The Cryptshare Server becomes unresponsive under high load.

Transfer Vorbereiten

Bevor ein Transfer agesendet werden kann muss dieser mit den folgenden Parametern vorbereitet werden:

- Kontaktdaten des Absenders
  - Name
  - Telefonnummer
  - E-Mail Adresse
- Das zu verwendende Passwortverfahren (Optional)
- Der Nachrichtentext für die Absenderbenachrichtung (Optional)
- Die Empfänger-E-Mail Adressen
- Die Sprache der Benachrichtigung (Optional)
- Die gewünschte Liegezeit für den Transfer (Optional)
- Die zu übertragenden Dateien

Wird der Transfer nicht gestattet da einige Empfänger dafür nicht zugelassen werden so wird eine Exception geworfen. Um dies zu vermeiden kann vor dem Transfer eine Policyüberprüfung durchgeführt werden.

Das Passwortverfahren

Manuelles Passwort - 'manual'

Ein Passwort muss explizit vom Absender eingegeben werden. Das Passwort kann mit den Funktionen aus dem Kapitel 'Passwortfunktionen' auf die aktuellen Sicherheitsanforderungen überprüft werden.
Der Empfänger muss bei diesem Verfahren Kontakt mit dem Absender aufnehmen um das Passwort zu erfahren.

Generiertes Passwort - 'generated'

Der Client fordert bei diesem Verfahren ein automatisch generiertes Passwort vom Server an welches die geforderten Sicherheitsanforderungen erfüllt.
Der Empfänger muss bei diesem Verfahren Kontakt mit dem Absender aufnehmen um das Passwort zu erfahren.

Kein Passwort - 'none'

Bei diesem Verfahren muss keiner der Beteiligten ein Passwort für den Transfer angeben. Dies wird vom Server für den Anwender 'unsichtbar' gehandhabt.
Beachten Sie bitte, dass dieser Modus als der unsichserste Modus gilt und daher nur in ausgewählten Fällen verwendet werden sollte.

Die Benachrichtigungssprache

Die Beanchrichtigungsprache kann für den Absender und die Empfänger separat über den entsprechenden ISO-639-1 Code angegeben werden. Sie können nur Sprachen angeben welche auf dem Server auch als Sprachpaket installiert sind (siehe Kapitel 'Sprach-Ressourcen').

Der Benachrichtigungtext

Soll für die Empfängerbenachrichtigung ein eigener Text verwendet werden so kann dieser über die Setter im Transfer-Objekt übergeben werden, entweder als Zeichenkette oder direkt als Stream. Der Text muss im UTF-8 Format angegeben werden und kann HTML-Markup enthalten.
Desweiteren kann auch ein eigener Betreff für die Benachrichtigung im Plaintext-Format übergeben werden.

Vertrauliche Nachrichten

Auf dieselbe Weise wie ein Benachrichtigungstext angegeben werden kann, kann auch eine vertrauliche Nachricht zu dem Transfer hinzugefügt werden. Diese wird zusammen mit den Transferdateien verschlüsselt und kann von den Empfängern beim Abruf des Transfers eingesehen werden. Ein optionaler Betreff für die vertrauliche Nachricht kann ebenfalls festgelegt werden.

Cryptshare Transfer vorbereiten

```
// URL zum Cryptshare Server
WebServiceUri uri = new WebServiceUri("https://cryptshare.yourdomain.com");

// Verbindungsinstanz zum Cryptshare Server
CryptshareConnection connection = new CryptshareConnection(uri);

// Client für das Absenden der Requests
Client client = new Client("john.adams@yourdomain.com", connection,
"C:\\temp\\client.store");

// Erzeugen eine Transfer Instanz
Transfer transfer = new Transfer();

// Festlegen des Absendernamens
transfer.setSenderName("John Adams");

// Festlegen der Absender-Telefonnummer
transfer.setSenderPhone("234 5467");

/**
 * Festlegen des Benachrichtigungstextes für die Empfängerbenachrichtigung.
 * Der Text kann direkt als UTF-8-codierter InputStream übergeben werden.
 **/
try {
   FileInputStream inputStream = new FileInputStream("C:\\temp\\message.txt");
   // Nachricht wird als InputStream übergeben. Der Stream wird automatisch
   // geschlossen.
   transfer.setMessage(inputStream);
} catch (Exception ex) {
   // Anzeigen einer Fehlernachricht bei Auftreten eines Fehlers
   System.err.println("Der Nachrichteninhalt konnte nicht gelesen werden!");
}

// Festlegen des Betreffs für die Empfängerbenachrichtigung
transfer.setSubject("Betreff für diesen Transfer");

// Festlegen der Empfänger
List<String> recipients = new ArrayList<String>();
```

```
recipients.add("jane.doe@abc.com");
recipients.add("jack.smith@xyz.com");

// Abrufen der Policy für diese Absender/Empfänger-Kombination um Sicher
// zu stellen, dass diese auch zugelassen sind.
Policy policy = client.requestPolicy(recipients);

// Nur zulässige Empfänger zum Transfer hinzufügen
if (policy.getFailedAddresses() != null && !policy.getFailedAddresses().isEmpty()) {
   for (String recipient : recipients) {
      if (!policy.getFailedAddresses().contains(recipient)) {
         transfer.addRecipient(recipient);
      } else {
         System.out.println("Der Empfänger ist unzulässig: " + recipient);
      }
   }
} else {
   // Alle Empfänger sind zugelassen.
   transfer.addRecipients(recipients);
}

// Sofern durch die Policy erlaubt sender wir außerdem eine Vertrauliche
// Nachricht
if (policy.isAllowConfidentialMessage()) {
   transfer.setConfidentialSubject("Betreff der vertraulichen Nachricht");
   transfer.setConfidentialMessage("Vertraulicher Nachrichtentext");
}
```

```
// Fahre nur dann fort wenn mindestens ein gültiger Empfänger vorhanden ist
if (transfer.getRecipients() == null || transfer.getRecipients().isEmpty())
   throw new IllegalStateException("Keine gültigen Empfänger gefunden. Der Transfer
   // abgebrochen.");

// Festlegen des Passwortverfahrens. In diesem Falle wird der als am
// sichersten geltende Modus verwendet.
PasswordMode passwordMode;
List<PasswordMode> allowedPasswordModes = policy.getPasswordMode();
if (allowedPasswordModes.contains(PasswordMode.GENERATED))
   passwordMode = PasswordMode.GENERATED;
else if (allowedPasswordModes.contains(PasswordMode.MANUAL))
   passwordMode = PasswordMode.MANUAL;
else
   passwordMode = PasswordMode.NONE;
```

Transfer durchführen

Synchron

Ist der Transfer, wie im vorigen Kapitel beschrieben, vorbereitet so kann dieser nun mittels der Methode **#performTransfer(Transfer, TransferUploadListener)** synchron bereitgestellt werden. Folgende Parameter müssen dabei angegeben werden:

- Der vorbereitete Transfer
- Ein TransferUploadListener zur Anzeige des Transferfortschritts

Da dies eine synchrone Operation ist, wird die Methode so lange 'blockieren' bis alle Dateien hochgeladen und der Transfer abgeschlossen ist.

Durchführen eines synchronen Transfers

```
// Schritt 1: Erzeugen einer Client-Instanz

// Anlegen der URL zu Ihrem Cryptshare Server
WebServiceUri serviceUri = new WebServiceUri("https://cryptshare.server.com");

// Erzeugen der Verbindung zum Cryptshare Server
CryptshareConnection connection = new CryptshareConnection(serviceUri);

// Erzeugen der Client-Instanz unter Verwendung der Absenderadresse,
// der Verbindung zum Server und des Pfades für denn lokalen Verifizierungsspeicher.
Client client = new Client("sender_email@server.com", connection, "C:\temp");

// Vorbereiten des Transferobjektes wie in vorigem Kapitel beschrieben.
Transfer transfer = ...

// Durchführen eines synchronen Transfers mit anonymen TransferUploadListener.
// Die Methode blockiert so lange bis der Transfer abgeschlossen ist.
client.performTransfer(transfer, new TransferUploadListener() {
```

```
// Festlegen des Datums an dem der Transfer abläuft.
// In diesem Falle wird das erlaubte maximum verwendet.
int storageDuration = policy.getStorageDuration();
Calendar cal = Calendar.getInstance();
cal.add(Calendar.DAY_OF_MONTH, storageDuration);
transfer.setExpirationDate(cal.getTime());
```

```
    @Override
    public void handleUploadProgressChanged(ProgressChangedEvent evt) {
        // Diese Methode wird während des Uploads wiederholt aufgerufen und gibt
        // den Transferfortschritt auf der Kommandozeile aus.
        double percent = (((double) evt.getBytesUploaded() / evt.getBytesTotal()) *
100.0);
        System.out.println("Transferfortschritt ... " + ((int)percent) + "%");
    }

    @Override
    public void handleUploadCompleted(UploadCompletedEvent evt) {
        // Diese Methode wird aufgerufen wenn alle Dateien auf den Server hochgeladen
wurden.
        System.out.println("Upload vollständig!");
    }

    @Override
    public void handleUploadInterrupted(UploadInterruptedEvent evt) {
        // Diese Methode wird aufgerufen wenn ein Fehler während des Uploads auftritt
        System.out.println("Während des Uploads ist ein Fehler aufgetreten: " +
evt.getException());
    }

    @Override
    public void handleTransferCanceled() {
        // Diese Methode wird aufgerufen wenn der Transfer
        // mittels #cancelTransfer() abgebrochen wurde.
        System.out.println("Der Transfer wurde vom Benutzer abgebrochen.");
    }
});
```

Asynchron

Ist der Transfer, wie im vorigen Kapitel beschrieben, vorbereitet so kann dieser nun mittels der Methode **#beginTransfer(Transfer, TransferUploadListener)** asynchron bereitgestellt werden. Folgende Parameter müssen dabei angegeben werden:

- Der vorbereitete Transfer
- Ein TransferUploadListener zur Anzeige des Transferfortschritts

Entgegen dem synchronen Upload wird diese Methode unmittelbar abgeschlossen nachdem der Transfer in einem eigenen Thread gestartet wurde. Ist der Transfer abgeschlossen so wird die Methode **TransferUploadListener#handleUploadCompleted(UploadCompletedEvent)** aufgerufen.

Durchführen eines asynchronen Transfers

```
// Schritt 1: Erzeugen einer Client-Instanz

// Anlegen der URL zu Ihrem Cryptshare Server
WebServiceUri serviceUri = new WebServiceUri("https://cryptshare.server.com");

// Erzeugen der Verbindung zum Cryptshare Server
CryptshareConnection connection = new CryptshareConnection(serviceUri);

// Erzeugen der Client-Instanz unter Verwendung der Absenderadresse,
// der Verbindung zum Server und des Pfades für denn lokalen Verifizierungsspeicher.
Client client = new Client("sender_email@server.com", connection, "C:\temp");

// Vorbereiten des Transferobjektes wie in vorigem Kapitel beschrieben.
Transfer transfer = ...

// Starten eines asynchronen Transfers mit anonymen TransferUploadListener.
// Die Methode wird unmittelbar nach Aufruf beendet.
client.beginTransfer(transfer, new TransferUploadListener() {
```

```
    @Override
    public void handleUploadProgressChanged(ProgressChangedEvent evt) {
        // Diese Methode wird während des Uploads wiederholt aufgerufen und gibt
        // den Transferfortschritt auf der Kommandozeile aus.
        double percent = (((double) evt.getBytesUploaded() / evt.getBytesTotal()) *
100.0);
        System.out.println("Transferfortschritt ... " + ((int)percent) + "%");
    }

    @Override
    public void handleUploadCompleted(UploadCompletedEvent evt) {
        // Diese Methode wird aufgerufen wenn alle Dateien auf den Server hochgeladen
wurden.
        System.out.println("Upload vollständig!");
    }

    @Override
    public void handleUploadInterrupted(UploadInterruptedEvent evt) {
        // Diese Methode wird aufgerufen wenn ein Fehler während des Uploads auftritt
        System.out.println("Während des Uploads ist ein Fehler aufgetreten: " +
evt.getException());
    }

    @Override
    public void handleTransferCanceled() {
        // Diese Methode wird aufgerufen wenn der Transfer
        // mittels #cancelTransfer() abgebrochen wurde.
        System.out.println("Der Transfer wurde vom Benutzer abgebrochen.");
    }
});
```

Cryptshare Java API Handbuch : E-Mail Benachrichtigungen

Created by René Hartwig, last modified on May 23, 2018

E-Mail Benachrichtigungen auf Serverseite deaktivieren

Für gewöhnlich verarbeitet der Cryptshare Server alle E-Mail Benachrichtigungen zwischen den Teilnehmern eines Cryptshare Transfers. Bei der Entwicklung einer eigenen Client-Anwendung für die Cryptshare-Kommunikation kann es jedoch erwünscht sein, die E-Mail-Benachrichtigung aus dem Client heraus durchzuführen.
Aus diesem Grund kann in der API die Benachrichtigung an Absender und Empfänger deaktiviert werden.

Absender-Benachrichtigung für einen Transfer deaktivieren

```
Transfer transfer = new Transfer();
transfer.setNotifySender(false);
```

Empfänger-Benachrichtigung für einen Transfer deaktivieren

```
Transfer transfer = new Transfer();
transfer.setNotifyRecipients(false);
```

E-Mail Templates vom Cryptshare Server abrufen

Bitte beachten Sie auch die Dokumentation zu Cryptshare Sprachpaketen im Cryptshare Server Handbuch.

Entwickler einer Cryptshare Client-Anwendung möchten die Transferbenachrichtigungen möglicherweise selber verwalten, beispielsweise für eine Client-Anwendung wie Cryptshare for Outlook. Um dennoch dasselbe Layout und insbesondere die spezifischen Informationen pro Empfänger zu erhalten gibt es in der API die Möglichkeit, den HTML-Code für die E-Mail Benachrichtigung vom Server zu erhalten.

```
client.requestMailTemplate("<template-name>", <replacements>, <language>, <mailFormat>);
```

<template-name> : Der Name des gewünschten Templates
<replacements> : Mapping der Platzhalter im Template
<language> : Die Sprache des Templates
<mailFormat> : Das E-Mail-Format welches verwendet werden soll. Entweder '**html**' oder '**plain**'.

Verwandte Knowledgebaseartikel

- KeyNotFoundException after confirming the transfer dialog of a Cryptshare transfer
- Error message "System Error. Code: 18." when performing Cryptshare transfer
- Error message "The path is not of a legal form." when performing Cryptshare transfer
- Performance issues with Outlook when using Cryptshare for Office 365 & Outlook
- Cryptshare for Outlook does not recognize the incoming verification e-mail
-

- Exception with message "LoadServerSettingsOnStartupAsync" is thrown when launching Outlook
- Office application crashes shortly after launching
- Outlook crashes when switching to a certain folder
- Error message "MAPI_E_INVALID_PARAMETER" when user performs a transfer
- Error message "Some components of Cryptshare for Outlook V2 don't support [...]" when launching Outlook with the Add-in
- Several "Get List Of All User Account Exception" errors in the log file
- AccessViolationException when adding attachments via drag and drop
- Shared mailbox is not recognized by the Add-in
- User receives "System.MissingMethodException" exception when launching an Office application
- The Cryptshare Server becomes unresponsive under high load.

E-Mail Platzhalter

E-Mail Platzhalter werden benutzt um individuelle Informationen in Templates auszufüllen. Dadurch kann jeder E-Mail Empfänger eine E-Mail mit spezifischem Inhalt, wie beispielsweise einem persönlichen Downloadlink, Name oder E-Mail Adresse, erhalten.

Template-Schnipsel mit Platzhaltern für Name und E-Mail Adresse

```
[...]
<p>Dear Sir or Madam,</p>

<p>Confidential data has been sent to you by <a href="mailto:$email">$name</a>.
[...]
```

Mögliche Platzhalter in E-Mail Benachrichtigungen

Die folgende Tabelle zeigt mögliche Platzhalter in E-Mail Benachrichtigungen. Platzhalter die mit `ABSENDER` markiert sind werden für das Absender-Template benötigt.
Platzhalter die mit `EMPFÄNGER` markiert sind werden für das Empfänger-Template benötigt.

Tag	Placeholder Key	Description	Additional Notes
ABSENDER EMPFÄNGER	`TemplatePlaceholder.BASEURL`	Die Cryptshare Server URL	`i.e. https://server.url.com`
EMPFÄNGER	`TemplatePlaceholder.SUBJECT`	Der Betreff der Nachricht	
EMPFÄNGER	`TemplatePlaceholder.MESSAGE`	Der Nachrichtentext	
ABSENDER	`TemplatePlaceholder.DATE_`	Das Ablaufdatum des Transfers	

EMPFÄNGER			
ABSENDER EMPFÄNGER	`TemplatePlaceholder.LIST_1`	Die Liste der Dateien in dem Transfer	
EMPFÄNGER	`TemplatePlaceholder.LINK`	Der Downloadlink für den Transfer	
ABSENDER EMPFÄNGER	`TemplatePlaceholder.PASSWORD_MODE`	Das für den Transfer verwendete Passwortverfahren	Mögliche Werte: • manual • generated • none
EMPFÄNGER	`TemplatePlaceholder.NAME`	Der Name des Absenders	
EMPFÄNGER	`TemplatePlaceholder.PHONE`	Die Telefonnummer des Absenders	
EMPFÄNGER	`TemplatePlaceholder.EMAIL`	Die E-Mail Adresse des Absenders	
ABSENDER EMPFÄNGER	`TemplatePlaceholder.LIST_2`	Liste der Empfänger die unter 'An' aufgeführt sind	
ABSENDER EMPFÄNGER	`TemplatePlaceholder.LIST_3`	Liste der Empfänger die unter 'Cc' aufgeführt sind	

Bitte beachten Sie, dass die obenstehende Tabelle lediglich die Standardkonfiguration für Cryptshare E-Mails zeigt. Jeder Platzhalter kann ebenfalls in eigenen Templates für eigene Zwecke umfunktioniert werden. Lesen Sie hierzu bitte die entsprechende Dokumentation im Cryptshare Server Handbuch.

Wie E-Mail Platzhalter verwendet werden können

#client.requestMailTemplate() verlangt ein sehr spezifisches Mapping damit man das gewünschte Resultat erhält. Der Parameter ist ein Set welches Instanzen des Typs *TemplateReplacement* hält. Ein *TemplateReplacement*-Objekt ist im Wesentlichen ein Key-Value Wrapper bei dem der Schlüssel ein spezifische Enum-Platzhalter ist und der Wert entweder eine Liste oder ein einzelner Wert ist. Auf diese Art können angepasste E-Mail Templates pro Empfänger mit einem einzigen Request angefordert werden.

Platzhalter verwenden

```
// <replacements>
Map<Set<String>, Set<TemplateReplacement>> replacements = new HashMap<Set<String>,
Set<TemplateReplacement>>();
// <replacementSet>
Set<TemplateReplacement> replacementSet = new HashSet<TemplateReplacement>();
replacementSet.add(new TemplateReplacement(TemplatePlaceholder.NAME, "John Adams"));
replacementSet.add(new TemplateReplacement(TemplatePlaceholder.EMAIL,
"john.adams@server.com"));

List<String> fileList = new ArrayList<String>();
fileList.add("file_01.txt");
fileList.add("file_02.docx");
replacementSet.add(new TemplateReplacement(TemplatePlaceholder.LIST_3, fileList));

List<String> recipientList = new ArrayList<String>();
recipientList.add("recipient1@otherdomain.org");
recipientList.add("recipienr2@otherdomain.com");
```

```
replacementSet.add(new TemplateReplacement(TemplatePlaceholder.LIST_1, recipientList));

// <identifiers>
Set<String> identifiers = new HashSet<String>();
identifiers.addAll(recipientList);
replacements.put(identifiers, replacementSet);

Map<List<String>, String> mailTemplates = client.requestMailTemplate("recipient",
replacements, new Locale("ja"), "html");
```

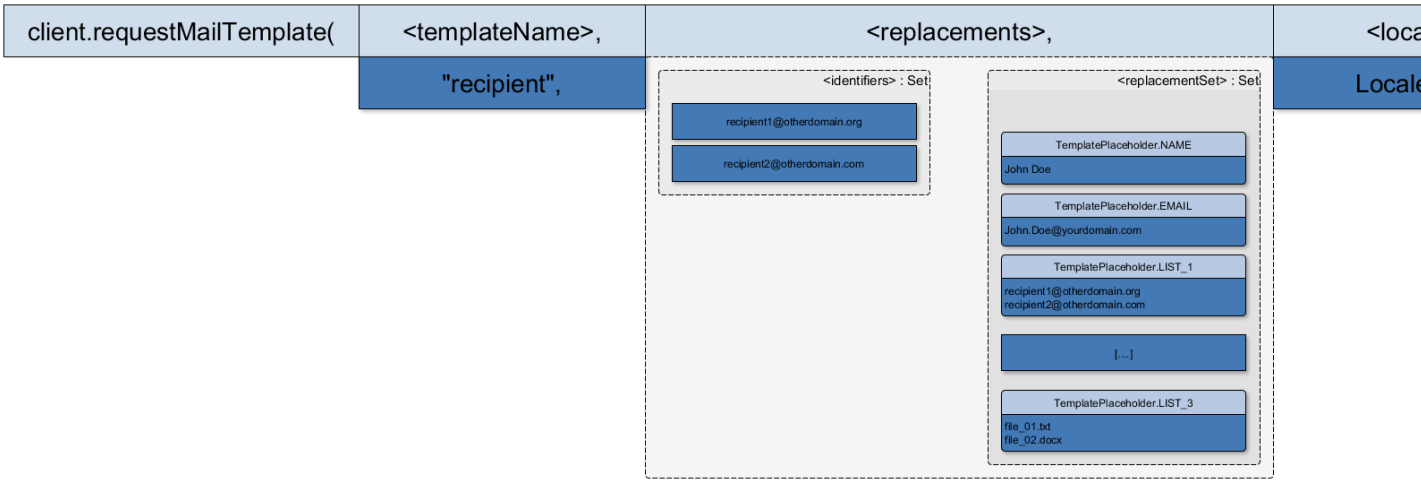
Das Ergebnisobjekt eines E-Mail-Template-Requests

Wird ein E-Mail Template mittels *client.requestMailTemplate()* angefordert enthält das zurückgegebene Mapping ein spezifisches Template pro Identifier-Set. Das bedeutet, dass ein ausgefülltes Template mit einem einzigen Request für mehrere Empfänger, einzelne Empfänger oder einer Kombination von Beidem abgerufen werden kann.

Beispiel: Mehrere Kombinationen von <identifier> und <replacementSet>

```
// identifiers1: One single recipient, replacementSet1: Set of replacements
replacements.put(identifiers1, replacementSet1);
// identifiers2: Two recipients, replacementSet2: Set of replacements
replacements.put(identifiers2, replacementSet2);
Map<List<String>, String> mailTemplates = client.requestMailTemplate("recipient",
replacements, new Locale("ja"), "html");
```

Der obenstehende Request würde zwei Templates zurückgeben, eines pro Identifier-Set das dem replacement-Parameter übergeben wurde. Beide Templates haben einen anderen Inhalt der mittels *replacementSet1*, bzw. *replacementSet2* ausgefüllt wurde.



<templateName> : Der Name des gewünschten Templates
<replacements> : Mapping zwischen <identifiers> und <replacementSet>
<identifiers> : Set aus E-Mail Adressen die ein spzifisches <replacementSet> haben
<replacementSet> : Ein Set von Platzhaltern Welche <identifiers> zugeordnet sind

Befine Solutions AG - Schwarzwaldstr. 151 - 79102 Freiburg - GERMANY
Technical Support: Phone +49 761 389 13 100 - E-Mail: support@cryptshare.com

Cryptshare Java API Handbuch : Lizenzinformationen

Created by René Hartwig, last modified on May 23, 2018

Mit der Methode *client#requestLicenseInfo()* können aktuelle Lizenzinformationen vom Server abgerufen werden. Dies kann nützlich sein um in der eigenen Client-Anwendung einen entsprechenden Hinweis einzublenden falls die Lizenz bald ablaufen sollte.
Die Methode gibt ein Objekt des Typs *LicenseInfo* zurück welches die folgenden Informationen beinhaltet:

- Ablaufdatum der Lizenz im Format 'YYYY-MM-DD'
- Ablaufdatum der Subskription im Format 'YYYY-MM-DD'
- Aktueller Status der Lizenz

Beispiel: Lizenzinformationen abrufen

```
// Client Instanz erzeugen
// WebServiceUri zum Cryptshare Server anlegen
WebServiceUri serviceUri = new WebServiceUri("https://cryptshare.server.com");

// CryptshareConnection-Instanz anlegen
CryptshareConnection connection = new CryptshareConnection(serviceUri);

// Client-Instanz mit Absenderadresse, CryptshareConnection und Verifizierungsspeicher
anlegen
Client client = new Client("sender_email@server.com", connection, "C:\temp");

// Request the License Information
LicenseInfo licenseInfo = client.requestLicenseInfo();

System.out.println("Ablaufdatum der Lizenz: " +
                        licenseInfo.getServerLicenseExpirationDate());
System.out.println("Ablaufdatum der Subskription: " +
                        licenseInfo.getServerSubscriptionExpirationDate());
System.out.println("Ist die Lizenz noch gültig? " + licenseInfo.isServerLicenseValid());
System.out.println("Ablaufdatum der Produktlizenz: " +
                        licenseInfo.getProductLicenseExpirationDate());
System.out.println("Ablaufdatum der Produktsubskription: " +
                        licenseInfo.getProductSubscriptionExpirationDate());
System.out.println("Ist die Produktlizenz noch gültig? " +
licenseInfo.isProductLicenseValid());
```

Verwandte Knowledgebaseartikel

- KeyNotFoundException after confirming the transfer dialog of a Cryptshare transfer
- Error message "System Error. Code: 18." when performing Cryptshare transfer
- Error message "The path is not of a legal form." when performing Cryptshare transfer
- Performance issues with Outlook when using Cryptshare for Office 365 & Outlook
-

- Cryptshare for Outlook does not recognize the incoming verification e-mail
- Exception with message "LoadServerSettingsOnStartupAsync" is thrown when launching Outlook
- Office application crashes shortly after launching
- Outlook crashes when switching to a certain folder
- Error message "MAPI_E_INVALID_PARAMETER" when user performs a transfer
- Error message "Some components of Cryptshare for Outlook V2 don't support [...]" when launching Outlook with the Add-in
- Several "Get List Of All User Account Exception" errors in the log file
- AccessViolationException when adding attachments via drag and drop
- Shared mailbox is not recognized by the Add-in
- User receives "System.MissingMethodException" exception when launching an Office application
- The Cryptshare Server becomes unresponsive under high load.

Befine Solutions AG - Schwarzwaldstr. 151 - 79102 Freiburg - GERMANY
Technical Support: Phone +49 761 389 13 100 - E-Mail: support@cryptshare.com

Cryptshare Java API Handbuch : Nutzungsbestimmungen

Created by René Hartwig, last modified on May 23, 2018

Die Nutzungsbestimmungen des Cryptshare Server können in der Administrationsoberfläche im Bereich 'Rechtliche Hinweise' definiert werden und vom Client heruntergeladen werden um Sie den Benutzern anzuzeigen. Hierfür ist die Methode **#requestTermsOfUse()** gedacht. Die zurückgelieferte Map enthält die Bestimmungen als String sowie die entsprechende Sprache.

Abrufen der Nutzungsbestimmungen

```
// Schritt 1: Erzeugen einer Client-Instanz

// Anlegen der URL zu Ihrem Cryptshare Server
WebServiceUri serviceUri = new WebServiceUri("https://cryptshare.server.com");

// Erzeugen der Verbindung zum Cryptshare Server
CryptshareConnection connection = new CryptshareConnection(serviceUri);

// Erzeugen der Client-Instanz unter Verwendung der Absenderadresse,
// der Verbindung zum Server und des Pfades für denn lokalen Verifizierungsspeicher.
Client client = new Client("sender_email@server.com", connection, "C:\temp");

// Abrufen der Nutzungsbestimmungen
Map<String,String> terms = client.requestTermsOfUse();

for (Entry<String,String> entry : terms.entrySet()) {
    System.out.println("Nutzungsbestimmungen in der Sprache '"
              + entry.getKey() + "': " + entry.getValue());
}
```

Verwandte Knowledgebase-Artikel

- KeyNotFoundException after confirming the transfer dialog of a Cryptshare transfer
- Error message "System Error. Code: 18." when performing Cryptshare transfer
- Error message "The path is not of a legal form." when performing Cryptshare transfer
- Performance issues with Outlook when using Cryptshare for Office 365 & Outlook
- Cryptshare for Outlook does not recognize the incoming verification e-mail
- Exception with message "LoadServerSettingsOnStartupAsync" is thrown when launching Outlook
- Office application crashes shortly after launching
- Outlook crashes when switching to a certain folder
-

- Error message "MAPI_E_INVALID_PARAMETER" when user performs a transfer
- Error message "Some components of Cryptshare for Outlook V2 don't support [...]" when launching Outlook with the Add-in
- Several "Get List Of All User Account Exception" errors in the log file
- AccessViolationException when adding attachments via drag and drop
- Shared mailbox is not recognized by the Add-in
- User receives "System.MissingMethodException" exception when launching an Office application
- The Cryptshare Server becomes unresponsive under high load.

Befine Solutions AG - Schwarzwaldstr. 151 - 79102 Freiburg - GERMANY
Technical Support: Phone +49 761 389 13 100 - E-Mail: support@cryptshare.com

Cryptshare Java API Handbuch : Passwortfunktionen

Created by René Hartwig, last modified on May 23, 2018

Für die Durchführung eines Cryptshare-Transfers wird ein Passwort benötigt welches die Passwortanforderungen des Servers erfüllen muss.
Um sicherzustellen, dass die Anforderungen erfüllt werden können Sie den Cryptshare Server ein eingegebenes Passwort überprüfen lassen, oder den Server ein gültiges Passwort generieren lassen.

Anfordern eines generierten Passwortes vom Server

Durch den Aufruf der Methode **Client#requestPassword(int)** können Sie vom Server ein neues Passwort anfordern. Um diese Methode verwenden zu können, muss der Client verifiziert sein (Siehe Abschnitt 'Verifizierung'). Die Methode fordert die Angabe der gewünschten Länge für das Passwort und liefert ein, mit den Passworteinstellungen des Servers kompatibles, Passwort zurück. Anschließend kann dieses Passwort für einen Cryptshare Transfer verwendet werden. Liegt die angegebene Passwortlänge unter der geforderten Mindestlänge, so wird die Mindestlänge verwendet.

Beispiel: Anfordern eines Passwortes

```
// Schritt 1: Erzeugen einer Client-Instanz
// Anlegen der URL zu Ihrem Cryptshare Server
WebServiceUri serviceUri = new WebServiceUri("https://cryptshare.server.com");

// Erzeugen der Verbindung zum Cryptshare Server
CryptshareConnection connection = new CryptshareConnection(serviceUri);

// Erzeugen der Client-Instanz unter Verwendung der Absenderadresse,
// der Verbindung zum Server und des Pfades für denn lokalen Verifizierungsspeicher.
Client client = new Client("sender_email@server.com", connection, "C:\temp");

// Schritt 2: Verwenden der Service-Methoden

// Vorausgesetzt eine gültige Verifizierung ist vorhanden,
// können wir nun ein Passwort, mit beispielsweise 8 Zeichen, anfordern.
String password = client.requestPassword(8);
```

Verwandte Knowledgebase-Artikel

- KeyNotFoundException after confirming the transfer dialog of a Cryptshare transfer
- Error message "System Error. Code: 18." when performing Cryptshare transfer
- Error message "The path is not of a legal form." when performing Cryptshare transfer
- Performance issues with Outlook when using Cryptshare for Office 365 & Outlook
- Cryptshare for Outlook does not recognize the incoming verification e-mail
- Exception with message "LoadServerSettingsOnStartupAsync" is thrown when launching Outlook

- Office application crashes shortly after launching
- Outlook crashes when switching to a certain folder
- Error message "MAPI_E_INVALID_PARAMETER" when user performs a transfer
- Error message "Some components of Cryptshare for Outlook V2 don't support [...]" when launching Outlook with the Add-in
- Several "Get List Of All User Account Exception" errors in the log file
- AccessViolationException when adding attachments via drag and drop
- Shared mailbox is not recognized by the Add-in
- User receives "System.MissingMethodException" exception when launching an Office application
- The Cryptshare Server becomes unresponsive under high load.

Überprüfen eines Passwortes

Wenn Sie ein eigenes Passwort für den Transfer verwenden möchten, so können Sie vom Server prüfen lassen, ob die Sicherheitsanforderungen für Passwörter erfüllt sind. Ist dies nicht der Fall, wird beim Versuch den Transfer mit diesem Passwort zu initiieren ein Fehler auftreten.
Die Methode **Client#checkPassword(String)** nimmt das zu verwendende Passwort als Parameter entgegen und lässt dieses vom Server validieren. Zurückgeliefert wird ein '**PasswordPolicy**'-Objekt welches das Ergebnis der Überprüfung enthält.

Beispiel=Passwort anfordern

```
// Schritt 1: Erzeugen einer Client-Instanz
// Anlegen der URL zu Ihrem Cryptshare Server
WebServiceUri serviceUri = new WebServiceUri("https://cryptshare.server.com");

// Erzeugen der Verbindung zum Cryptshare Server
CryptshareConnection connection = new CryptshareConnection(serviceUri);

// Erzeugen der Client-Instanz unter Verwendung der Absenderadresse,
// der Verbindung zum Server und des Pfades für denn lokalen Verifizierungsspeicher.
Client client = new Client("sender_email@server.com", connection, "C:\temp");

// Schritt 2: Verwenden der Service-Methoden

// Vorausgesetzt eine gültige Verifizierung ist vorhanden, können wir nun,
// beispielsweise das Passwort 'password123' überprüfen lassen.
String passwordToCheck = "password123";
PasswordPolicy passwordPolicy = client.checkPassword(passwordToCheck);

// Wie sicher das Passwort eingestuft wird kann als float-Wert auf einer Skala von
// 0.0f - 1.0f mit 1.0f als 'Sehr Sicher' abgefragt werden.
System.out.println("password security = " + passwordPolicy.getSecurity());

// Das PasswordPolicy-Objekt enthält außerdem die Resultate der Sicherheitsanforderungen
// des übergebenen Passwortes
System.out.println("Ist das Passwort zu kurz?: " + passwordPolicy.isTooShort());
System.out.println("It das passwort zu lang?: " + passwordPolicy.isTooLong());
```

```
System.out.println("Erfüllt das Passwort nicht alle Anforderungen?: " +
                           passwordPolicy.isInsufficientCharacteristics());
System.out.println("Wurden im Passwort keine Ziffern verwendet?: " +
                           passwordPolicy.isInsufficientDigits());
System.out.println("Wurden im Passwort alphabetische Zeichen verwendet?: " +
                           passwordPolicy.isInsufficientAlphabetical());
System.out.println("Wurden im Passwort keine Sonderzeichen verwendet?: " +
                           passwordPolicy.isInsufficientSpecial());
System.out.println("Wurden im Passwort keine Großbuchstaben verwendet?: " +
                           passwordPolicy.isInsufficientUpper());
System.out.println("Wurden im Passwort keine Kleinbuchstaben verwendet?: " +
                           passwordPolicy.isInsufficientLower());
System.out.println("Enthält das Passwort Whitespaces (Leerzeichen, Umbrüche, etc.)?: " +

                           passwordPolicy.isIllegalWhitespace());
System.out.println("Ist das Passwort ein Wort?: " +
                           passwordPolicy.isIllegalWord());
System.out.println("Enthält das Passwort Sequenzen (123, abc, etc.)?: " +
                           passwordPolicy.isIllegalSequence());
System.out.println("Enthält das Passwort unerlaubte Wiederholungen (aaa, 111, etc.)?: "
+
                           passwordPolicy.isIllegalRepetition());

// Das PasswordPolicy-Objekt enthält außerdem Informationen darüber, wie die
// Passwortanforderungen am Server konfiguriert sind.
System.out.println("Minimale Passwortlänge: " + passwordPolicy.getMinimumLength());
System.out.println("Maximale Passwortlänge: " + passwordPolicy.getMaximumLength());
System.out.println("Muss ein Passwort Ziffern enthalten?: " +
                           passwordPolicy.isMustContainDigits());
System.out.println("Muss ein Passwort reguläre Buchstaben enthalten?: " +
                           passwordPolicy.isMustContainChars());
System.out.println("Muss ein Passwort Sonderzeichen enthalten?: " +
                           passwordPolicy.isMustContainSpecialChars());
System.out.println("Muss ein Passwort Groß/Kleinschreibung enthalten?: " +
                           passwordPolicy.isMustBeUpperLowerCase());
System.out.println("Werden reguläre Wörter nicht gestattet?: " +
                           passwordPolicy.isDictionaryDeclined());
System.out.println("Werden Passwörter mit Zeichenwiederholungen abgelehnt?: " +
                           passwordPolicy.isCharRepetitionsDeclined());
System.out.println("Sind Whitespaces (Leerzeichen, Umbrüche, etc.) erlaubt?: " +
                           passwordPolicy.isAllowWhitespaces());
System.out.println("Sind alphabetische Sequenzen gestattet?: " +
                           passwordPolicy.isAllowAlphabeticalSequence());
System.out.println("Sind numerische Sequenzen gestattet?: " +
                           passwordPolicy.isAllowNumericSequence());
System.out.println("Sind Tastatursequenzen (qwerty) gestattet?: " +
                           passwordPolicy.isAllowQwertySequence());
```

Cryptshare Java API Handbuch : Policyregeln

Created by René Hartwig, last modified on May 23, 2018

Cryptshare verwendet Policyregeln um die Berechtigungen sowie verfügbaren Optionen bei einem Cryptshare Transfer zu steuern. Zum Abruf der in Frage kommenden Regeln kann die Policy für eine bestimmte Absender/Empfänger Kombination mittels **#requestPolicy(List<String>)** abgerufen werden. Folgende Parameter müssen der Methode übergeben werden:

- Liste der der Empfänger-E-Mail Adressen

Beachten Sie, dass diese Operation nur möglich ist wenn der Client verifiziert ist (siehe Kapitel Verifizierung).

Abrufen der Policyeinstellungen

```
// Schritt 1: Erzeugen einer Client-Instanz

// Anlegen der URL zu Ihrem Cryptshare Server
WebServiceUri serviceUri = new WebServiceUri("https://cryptshare.server.com");

// Erzeugen der Verbindung zum Cryptshare Server
CryptshareConnection connection = new CryptshareConnection(serviceUri);

// Erzeugen der Client-Instanz unter Verwendung der Absenderadresse,
// der Verbindung zum Server und des Pfades für denn lokalen Verifizierungsspeicher.
Client client = new Client("sender_email@server.com", connection, "C:\temp");

// Liste der Empfänger an die versendet werden soll
List<String> recipients = new ArrayList<String>();
recipients.add("john.smith@abc.com");
recipients.add("jane.adams@xyz.com");
Policy policy = client.requestPolicy(recipients);

/**
* Das erhaltene Policy-Objekt enthält die Einstellungen welche
* für die gewählte Absender/Empfänger-Kombination zutreffen
**/
System.out.println("Wird der Absender über Transferdownloads benachrichtigt? " +
                policy.isDownloadNotification());
System.out.println("Werden Dateinamen in E-Mail Benachrichtigungen angezeigt? " +
                policy.isShowFilenames());
System.out.println("Dürfen vertrauliche Nachrichten versandt werden?: " +
                policy.isAllowConfidentialMessage());
System.out.println("Die maximale Liegezeit der Dateien in Tagen: " +
                policy.getStorageDuration());
System.out.println("Das maximale Transfervolumen in MB: " +
                policy.getTransferLimit());
System.out.println("Empfänger die für den Transfer nicht zugelassen sind:" +
                policy.getFailedAddresses());
System.out.println("Die erlaubten Passwortverfahren: " +
                policy.getPasswordMode());
```

Verwandte Knowledgebase-Artikel

- KeyNotFoundException after confirming the transfer dialog of a Cryptshare transfer
- Error message "System Error. Code: 18." when performing Cryptshare transfer

- Error message "The path is not of a legal form." when performing Cryptshare transfer
- Performance issues with Outlook when using Cryptshare for Office 365 & Outlook
- Cryptshare for Outlook does not recognize the incoming verification e-mail
- Exception with message "LoadServerSettingsOnStartupAsync" is thrown when launching Outlook
- Office application crashes shortly after launching
- Outlook crashes when switching to a certain folder
- Error message "MAPI_E_INVALID_PARAMETER" when user performs a transfer
- Error message "Some components of Cryptshare for Outlook V2 don't support [...]" when launching Outlook with the Add-in
- Several "Get List Of All User Account Exception" errors in the log file
- AccessViolationException when adding attachments via drag and drop
- Shared mailbox is not recognized by the Add-in
- User receives "System.MissingMethodException" exception when launching an Office application
- The Cryptshare Server becomes unresponsive under high load.

Befine Solutions AG - Schwarzwaldstr. 151 - 79102 Freiburg - GERMANY
Technical Support: Phone +49 761 389 13 100 - E-Mail: support@cryptshare.com

Cryptshare Java API Handbuch : Sprach-Ressourcen

Created by René Hartwig, last modified on May 23, 2018

Der Cryptshare Server kann mit Sprachpaketen für den Client ausgestattet werden welche für das Benutzerinterface verwendet werden können. Dadurch ist eine zentrale Verwaltung der Sprachpakete für alle Clients möglich.

Verfügbare Sprachen

Um herauszufinden welche Sprachen am Server verfügbar sind können mit der Methode **#requestLanguagePacks()** Informationen über installierte Sprachpakete abgerufen werden. Mithilfe dieser Informationen können anschließend bestimmte Sprachpakete angefordert werden.

```
// Schritt 1: Erzeugen einer Client-Instanz

// Anlegen der URL zu Ihrem Cryptshare Server
WebServiceUri serviceUri = new WebServiceUri("https://cryptshare.server.com");

// Erzeugen der Verbindung zum Cryptshare Server
CryptshareConnection connection = new CryptshareConnection(serviceUri);

// Erzeugen der Client-Instanz unter Verwendung der Absenderadresse,
// der Verbindung zum Server und des Pfades für denn lokalen Verifizierungsspeicher.
Client client = new Client("sender_email@server.com", connection, "C:\temp");

// Nun kann eine Liste der installierten Sprachen angefordert werden
List<LanguagePack> languagePackList = client.requestLanguagePacks();

for (LanguagePack languagePack : languagePackList) {
    Locale locale = languagePack.getLocale();
    String version = languagePack.getLanguagePackVersion();
    long lastUpdate = languagePack.getLastUpdate();
    System.out.println("Language pack language = " + locale.getLanguage() +
            " with version = " + version + " last updated at " + lastUpdate);
}
```

Verwandte Knowledgebase-Artikel

- KeyNotFoundException after confirming the transfer dialog of a Cryptshare transfer
- Error message "System Error. Code: 18." when performing Cryptshare transfer
- Error message "The path is not of a legal form." when performing Cryptshare transfer
- Performance issues with Outlook when using Cryptshare for Office 365 & Outlook
- Cryptshare for Outlook does not recognize the incoming verification e-mail
- Exception with message "LoadServerSettingsOnStartupAsync" is thrown when launching Outlook

- Office application crashes shortly after launching
- Outlook crashes when switching to a certain folder
- Error message "MAPI_E_INVALID_PARAMETER" when user performs a transfer
- Error message "Some components of Cryptshare for Outlook V2 don't support [...]" when launching Outlook with the Add-in
- Several "Get List Of All User Account Exception" errors in the log file
- AccessViolationException when adding attachments via drag and drop
- Shared mailbox is not recognized by the Add-in
- User receives "System.MissingMethodException" exception when launching an Office application
- The Cryptshare Server becomes unresponsive under high load.

Sprachpaket herunterladen

Nun da die verfügbaren Sprachpakete bekannt sind, können bestimmte Inhalte aus dem Paket mittels der Methode **#requestLanguagePackFile(String,Locale)** angefordert werden. Die Methode erwartet die folgenden Parameter:

- Dateiname der gewünschten Datei mit Angabe von Verzeichnissen, falls vorhanden
- Gewünschte Sprache der angeforderten Datei

Beachten Sie, dass bei der Angabe des Dateinamens nur der **Name ohne Sprachkürzel** angegeben werden muss.

Beispiel: Anfordern der Sprachdatei 'lang_en_GB.xml'

```
Locale locale = Locale.UK;
byte[] langFileBytes = client.requestLanguagePackFile("lang.xml", locale);
```

```
// Schritt 1: Erzeugen einer Client-Instanz

// Anlegen der URL zu Ihrem Cryptshare Server
WebServiceUri serviceUri = new WebServiceUri("https://cryptshare.server.com");

// Erzeugen der Verbindung zum Cryptshare Server
CryptshareConnection connection = new CryptshareConnection(serviceUri);

// Erzeugen der Client-Instanz unter Verwendung der Absenderadresse,
// der Verbindung zum Server und des Pfades für denn lokalen Verifizierungsspeicher.
Client client = new Client("sender_email@server.com", connection, "C:\temp");

// Liste der verfügbaren Sprachen anfordern
List<LanguagePack> languagePackList = client.requestLanguagePacks();

/**
* Nun kann eine bestimmte Datei aus dem Sprachpaket angefordert werden mit
* dem Dateinamen (ohne Lokalisierung) als erstem Parameter und als zweiten
* Parameter die gewünschte Sprache.
**/
byte[] langFileBytes = client.requestLanguagePackFile("lang.xml",
                                        languagePackList.get(0).getLocale());
```

```
// Speichern der Datei auf der lokalen Festplatte
if (langFileBytes != null) {
    FileOutputStream outStream = null;
    try {
        outStream = new FileOutputStream("lang.xml");
        outStream.write(langFileBytes);
    }
    catch (Exception e) {
        e.printStackTrace();
    }
    finally {
        try {
            outStream.close();
        }
        catch (Exception e) {}
    }
}
```

Cryptshare Java API Handbuch : Transfer-Polling

Created by René Hartwig, last modified on May 23, 2018

Sie können die Methode #requestActiveTransfers() der Java API nutzen um eine Liste der aktiven Transfers, welche mit der E-Mail Adresse dieses Clients versandt wurden, abzurufen. Hierfür muss der verwendete Client verifiziert sein. Die Methode gibt eine Map zurück mit den Meta-IDs als Schlüssel und den Download-URLs (ohne Passwort) als Wert.
Eine Download-URL hat das folgende Format, wobei der Parameter 'id' für die Meta-ID des Transfers steht:
https://cryptshare.server.com/download1.php?id=33d03d8d6b

Beispiel: Abrufen eines aktiven Transfers

```
// Anlegen einer Client-Instanz
// Erzeugen einer WebServiceUri zum Cryptshare-Server
WebServiceUri serviceUri = new WebServiceUri("https://cryptshare.server.com");

// Erzeugen einer CryptshareConnection-Instanz unter Verwendung der WebServiceUri
CryptshareConnection connection = new CryptshareConnection(serviceUri);

// Anlegen einer Client-Instanz mit der E-Mail Adresse des Empfängers für den aktive
Transfers aberufen werden sollen.
Client client = new Client("John.Doe@server.com", connection, "C:\temp");

// Abrufen der aktiven Transfers
// Gibt eine Map mit den Transfer-IDs und Download-URLS der aktiven Transfers für diesen
Client zurück.
// (In diesem Beispiel: John.Doe@server.com)
// Diese Methode wirft eine Exception wenn die E-Mail Adresse des Clients nicht
verifiziert ist.
Map<String, String> transferIdMap = client.requestActiveTransfers();

for (Entry<String, String> entry : transferIdMap.entrySet()) {
    System.out.println("Eintrag in Transfer-ID Map : metaId = " + entry.getKey() + " url
= " + entry.getValue());
}
```

Verwandte Knowledgebaseartikel

- KeyNotFoundException after confirming the transfer dialog of a Cryptshare transfer
- Error message "System Error. Code: 18." when performing Cryptshare transfer
- Error message "The path is not of a legal form." when performing Cryptshare transfer
- Performance issues with Outlook when using Cryptshare for Office 365 & Outlook
- Cryptshare for Outlook does not recognize the incoming verification e-mail
- Exception with message "LoadServerSettingsOnStartupAsync" is thrown when launching Outlook
-

- Office application crashes shortly after launching
- Outlook crashes when switching to a certain folder
- Error message "MAPI_E_INVALID_PARAMETER" when user performs a transfer
- Error message "Some components of Cryptshare for Outlook V2 don't support [...]" when launching Outlook with the Add-in
- Several "Get List Of All User Account Exception" errors in the log file
- AccessViolationException when adding attachments via drag and drop
- Shared mailbox is not recognized by the Add-in
- User receives "System.MissingMethodException" exception when launching an Office application
- The Cryptshare Server becomes unresponsive under high load.

Befine Solutions AG - Schwarzwaldstr. 151 - 79102 Freiburg - GERMANY
Technical Support: Phone +49 761 389 13 100 - E-Mail: support@cryptshare.com

Cryptshare Java API Handbuch : API Java Doc

Created by René Hartwig, last modified on May 23, 2018

Befine Solutions AG - Schwarzwaldstr. 151 - 79102 Freiburg - GERMANY
Technical Support: Phone +49 761 389 13 100 - E-Mail: support@cryptshare.com

Cryptshare Java API Handbuch : Beispielprojekt

Created by Simon Erhardt, last modified by René Hartwig on Oct 30, 2018

Allgemeines

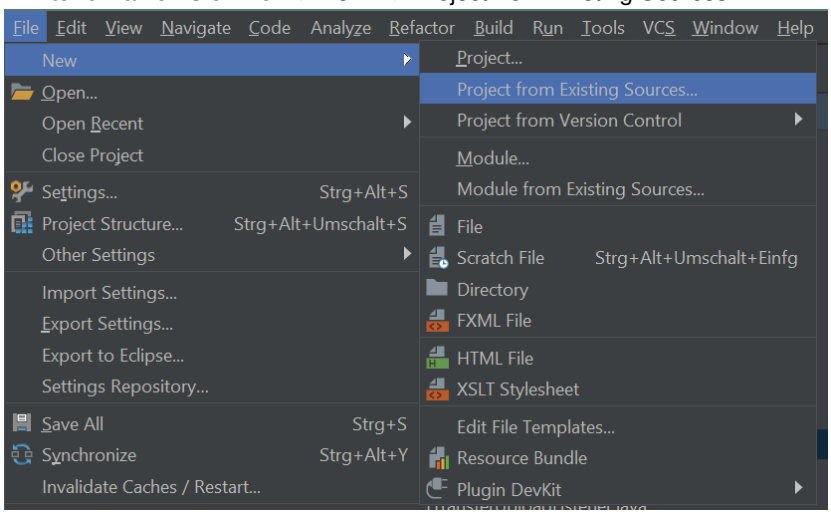
Diese Seite enthält ein einfaches Java Projekt, welches zeigt wie die Java API für Cryptshare verwendet werden kann.
Java API Version 3.0.0
Bitte beachten Sie, dass die Mavenkonfiguration zur Verwendung der API auf Version 3.0.0 eingestellt ist. Stellen Sie sicher, dass die entsprechende API Version in dem von Ihnen verwendeten Repository zur Verfügung steht.

Wie das Projekt in die bevorzugte IDE importiert werden kann

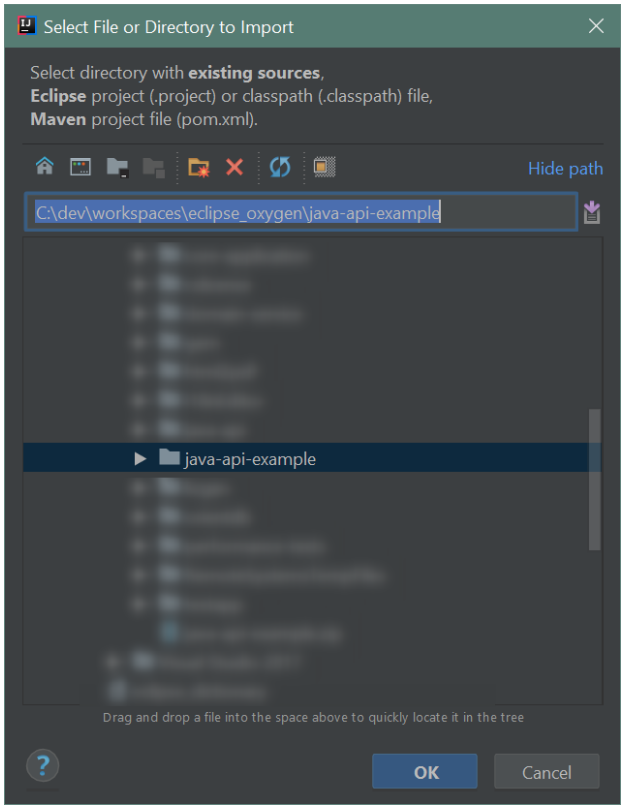
IntelliJ

Klicken Sie hier um die Anweisungen für den Import in IntelliJ anzuzeigen

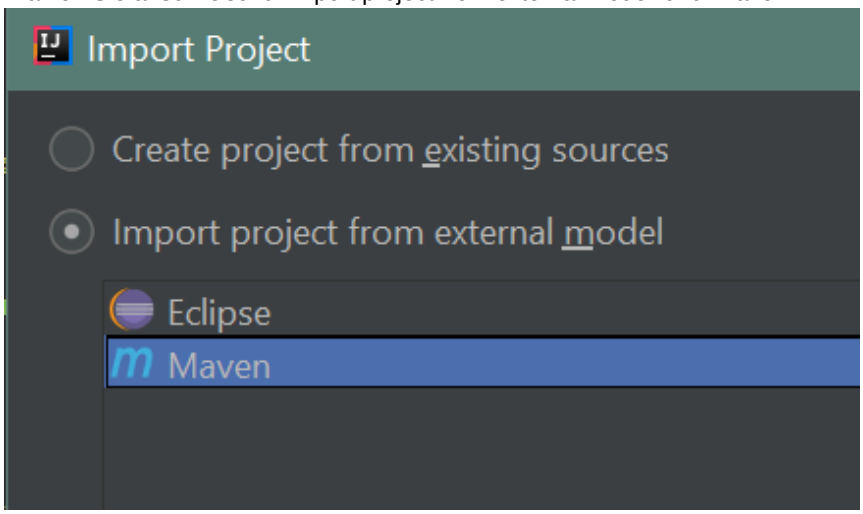
1. Laden Sie die Zipdatei mit dem Projekt herunter und entpacken Sie den Inhalt in Ihrem Workspace.
2. In IntelliJ wählen Sie 'File' → 'New' → 'Project from Existing Sources'



3. Wählen Sie anschließend das Verzeichnis für das Beispielprojekt aus



4. Wählen Sie anschließend 'Import project from external model' und 'Maven'.

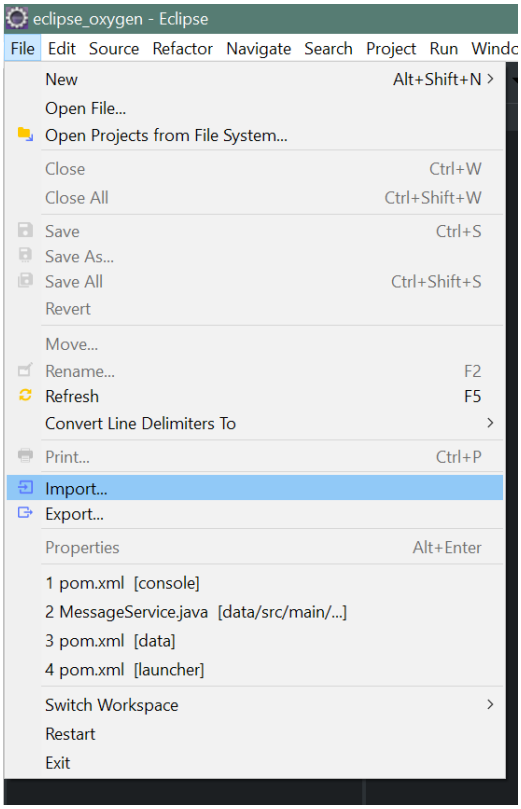


5. Folgen Sie den weiteren Schritten des Assistenten durch klicken auf 'Next'

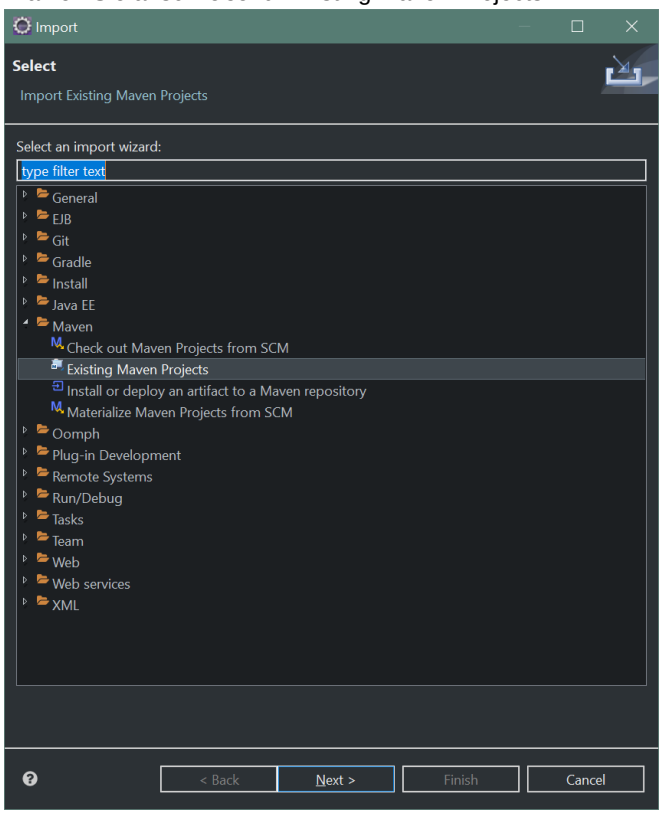
Eclipse

Klicken Sie hier um die Anweisungen für den Import in Eclipse anzuzeigen

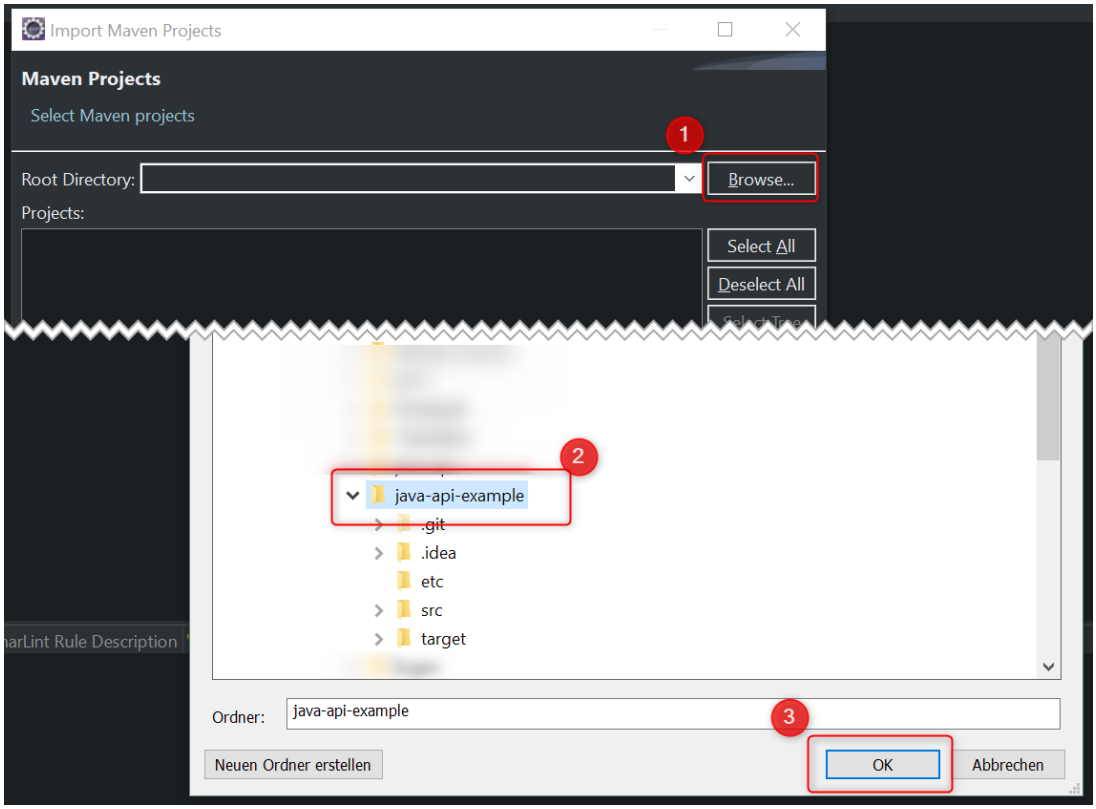
1. Laden Sie die Zipdatei mit dem Projekt herunter und entpacken Sie den Inhalt in Ihrem Workspace.
2. In Eclipse wählen Sie 'File' → 'Import'.



3. Wählen Sie anschließend 'Existing Maven Projects'.



4. Klicken Sie auf 'Browse', wählen Sie das Projektverzeichnis aus und klicken Sie auf 'OK'.



5. Klicken Sie auf 'Finish'.

Cryptshare Java API Handbuch : Kombatibilität

Created by Dirk Riesterer on Oct 30, 2018

Auf dieser Seite sehen sie die Übersicht über die Kombatibilität der Java API mit dem Cryptshare Server.

Cryptshare Server Version	Java API V2	Java API v3.0
3.12.0	✔	✔
3.12.1	✔	✔
4.0.0	✔	✔
4.1.0	✔	✔
4.1.1	✔	✔
4.1.2	✔	✔
4.1.3	✔	✔

Related Knowledge Base Articles

Content by label
There is no content with the specified labels