Webservice «Barcode»

Benutzerhandbuch Client API

Version 2.2
November 2013

DIE POST

Inhaltsverzeichnis

Referenzen

[1] Webservice «Barcode», Handbuch, http://www.post.ch/post-barcode-handbuch.pdf

[2] Dokumentationen zu Webservice «Barcode», http://www.post.ch/post-barcode-cug.htm

1 Einleitung

Diese Dokumentation erklärt die Anbindung des Webservice «Barcode», Version 2.2, unter Verwendung der Client API Libraries für Java und C# / .NET. An einigen typischen Beispielen wird die Verwendung der Libraries zum Zugriff auf den Webservice erklärt. Die Beispiele sind jeweils in C# und Java aufgeführt.

Diese Dokumentation ist weder eine vollständige Client-API-Dokumentation noch eine vollständige Beschreibung des Webservice.

1.1 Dokumentation Client API Webservice «Barcode»

Für eine vollständige Dokumentation der gesamten Client API Library verweisen wir auf die Source-Code-Dokumentation der Libraries, die als HTML-Dokumentation in der Distribution beiliegen.

- **Java:** Die Javadoc-API-Dokumentation befindet sich im Verzeichnis **doc,** die Sources sind im Source-ZIP-Archiv enthalten.
- **C#/.NET:** Die C#-API-Dokumentation befindet sich im Verzeichnis **doc,** die Sources sind im Source-ZIP-Archiv im Verzeichnis **src** enthalten.

1.2 Webservice-Dokumentation

Für eine vollständige Dokumentation des Webservice «Barcode» verweisen wir auf das Webservice-Benutzerhandbuch [1]. Dort werden auch sämtliche Begriffe, Attribute, Dienstleistungscodes, usw. erklärt. Weitere Ressourcen und Hilfsmittel zum Webservice «Barcode» finden Sie auf der Webservice-Barcode-Webseite [2].

1.3 Beispiel-Sources

Die Beispiele zur Verwendung der Client Library liegen ebenfalls als Sources der Library Distribution bei, und zwar im Verzeichnis **examples.**

2 Konfiguration

Das folgende Beispiel zeigt die einfache Initialisierung des Webservice «Barcode».

```
== Java Code: ==

BarcodeWebService service = new BarcodeWebService(Language.DE);
service.setLoginCredentials("your username", "your password");

// Now the web service operations can be called on the service instance.
```

```
== C# Code: ==

BarcodeWebService service = new BarcodeWebService(Language.de);
service.UserName = "your username";
service.Password = "your password";

// Now the web service operations can be called on the service instance.
```

Im Konstruktor des Webservice «Barcode» wird die gewünschte Sprache für allfällige Fehler- und Warnmeldungen vom Server ausgewählt. Folgende Einstellungen sind ausserdem zwingend vorzunehmen:

- **setLoginCredentials – Basic Authentication:** Username und Passwort für die Authentifizierung mit dem Webservice (Bemerkung: In C# werden diese Werte über die entsprechenden Eigenschaften **UserName** und **Password** gesetzt)
- **setFrankingLicense – Frankierlizenz:** Eine gültige Frankierlizenz für das Generieren von Barcodeetiketten. Diese Einstellung ist allerdings nur für Aufrufe der **generateLabel-**Operationen zwingend nötig
  (Bemerkung: In C# wird die Frankierlizenz über die Eigenschaft **FrankingLicense** gesetzt)

Nach der Instanziierung eines Webservice-Barcode-Objekts können über entsprechende Setter-Methoden diverse weitere Einstellungen des Webservice einfach vorgenommen werden. Folgende Einstellungen stehen zur Verfügung und sind bereits mit sinnvollen Voreinstellungen vorinitialisiert. Wird eine Einstellung nicht explizit gesetzt, so werden die voreingestellten Standardwerte verwendet.

Einstellungs-Property	Bedeutung	Voreingestellter Standardwert
WebserviceEndpointURL	Die URL, unter der der Webservice angesprochen werden soll. Diese Einstellung kann zum Beispiel für Testzwecke mit einem speziellen Testserver nützlich sein.	Produktive URL des Webservice «Barcode»: https://wsbc.post.ch/wsbc/barcode/v2_2
LabelLayout	Layout der zu erstellenden Adressetiketten: A5, A6, A7 oder FE	A6
ImageFileType	Bild- oder Dateiformat der Adressetiketten: PNG, GIF, PDF, SPDF, EPS, ZPL2 oder JPG	PNG
ImageResolution	Auflösung der Adressetiketten in DPI: 200, 300 oder 600	300
PrintAddresses	Einstellung, welche Adressen auf den Adressetiketten gedruckt werden sollen: RECIPIENT_AND_CUSTOMER (Empfänger und Absender), ONLY_RECIPIENT (nur Empfänger), NONE (keine)	RECIPIENT_AND_CUSTOMER
PPFranking	Gibt an, ob der Vermerk «PP Frankierung» auf dem Label aufgedruckt werden soll.	False
CustomerSystem	Dient dazu das Kundensystem des WSBC Services zu identifizieren. Ermöglicht einfacheres Nachvollziehen bei Problemen.	null

2.1 Proxykonfiguration

Falls der Internetzugang über einen Proxy erfolgt, muss dieser entsprechend konfiguriert werden.

In den folgenden Abschnitten ist diese Konfiguration für Java und C# beschrieben.

2.1.1 Proxykonfiguration in Java

Der Proxy kann in Java global über System-Properties konfiguriert werden. Folgende System-Properties können einfach über entsprechende VM-Argumente gesetzt werden:

System-Property	Beschreibung	Beispiel für VM-Argument
proxyHost	Die URL des Web-Proxy-Hosts	– DproxyHost="proxy.mycompany.com"
proxyPort	Der Port des Web-Proxies	– DproxyPort=8080
proxyUser	Username für Authentifizierung mit dem Web-Proxy	– DproxyUser="myusername"
proxyPassword	Passwort für Authentifizierung mit dem Web-Proxy	– DproxyPassword="mypassword"

Es ist auch möglich, diese System-Proxy-Einstellungen gemäss folgendem Beispiel programmatisch zu setzen.

```
== Java Code: ==

// Example: Setting proxy "proxy.mycompany.com" on port 8080
System.setProperty("proxyHost", "proxy.mycompany.com");
System.setProperty("proxyPort", "8080");

// Optional: username and password for web proxy that requires authentication
System.setProperty("proxyUser", "your proxy username");
System.setProperty("proxyPassword", "your proxy password");
```

Weitere Informationen zur Proxykonfiguration in Java finden Sie unter http://java.sun.com/javase/6/docs/technotes/guides/net/proxies.html.

2.1.2 Proxykonfiguration in C#/.NET

In C# kann die Proxykonfiguration direkt auf der Klasse WebRequest vorgenommen werden.

```
== C# Code: ==

// Example: Setting proxy "proxy.mycompany.com" on port 8080
WebProxy wProxy = new WebProxy("http://proxy.mycompany.com:8080");
WebRequest.DefaultWebProxy = wProxy;

// Optional: username and password for web proxy that requires authentication
NetworkCredential credentials =
    new NetworkCredential("myProxyUsername", "myProxyPassword");
WebRequest.DefaultWebProxy.Credentials = credentials;
```

Dem Konstruktor der BarcodeWebService Klasse kann zusätzlich ein Parameter übergeben werden, ob der Default Proxy des Betriebssystem (kann im Internet Explorer konfiguriert werden, wird in der Registry gespeichert) verwendet wird:

```
== C# Code: ==

// Use Proxy (configured by OS or manually configured)
BarcodeWebService service = new BarcodeWebService(Language.de, true);

// Don't use any proxy (default)
BarcodeWebService service = new BarcodeWebService(Language.de, false);
```

Der Parameter ist optional, der Default ist false. Bei einem produktiven Einsatz und Verwendung eines Proxys wird empfohlen diesen manuell zu konfigurieren. Die Verwendung des Default Proxy des Betriebssystems kann zu längeren Antwortzeiten führen.

3 Lese-Operationen

Mit den Lese-Operationen können alle aktuell unterstützten Dienstleistungscodes und Labellayouts des Webservice abgefragt werden. Alle Lese-Operationen werfen eine **BarcodeErrorException,** falls der Webservice-Barcode-Aufruf mit Fehlermeldungen (in der gewählten Sprache) vom Server beantwortet wurde.

Die folgenden Kapitel enthalten jeweils ein kurzes Beispiel für den Aufruf jeder Lese-Operation.

3.1 Operation «Lese Dienstleistungsgruppen»

Mit der Operation **readServiceGroups** können die momentan verfügbaren Dienstleistungsgruppen mit entsprechender **serviceGroupID** abgefragt werden.

== Java Code: ==

```
BarcodeWebService service = new BarcodeWebService(Language.DE);
service.setLoginCredentials("username", "passwort");

try {
    List<ServiceGroup> groups = service.readServiceGroups();
    System.out.println("Available Service-Groups:");
    for (ServiceGroup group : groups) {
        System.out.print(group.getServiceGroupID() + ": ");
        System.out.println(group.getDescription());
    }
} catch (BarcodeErrorException e) {
    System.out.println("Error in call to readServiceGroups:");
    System.out.println(e.getErrorMessages());
}
```

== C# Code: ==

```
BarcodeWebService service = new BarcodeWebService(Language.de);
service.UserName = "username";
service.Password = "password";

try
{
    List<PostLogistics.WebServiceBarcode.Client.ServiceGroup> groups =
service.ReadServiceGroups();
    Console.WriteLine("Available Service-Groups:");
    foreach (PostLogistics.WebServiceBarcode.Client.ServiceGroup group in groups)
    {
        Console.WriteLine("{0}: {1}", group.ServiceGroupID,
            group.Description);
    }
}
catch (BarcodeErrorException ex)
{
    Console.WriteLine("Error in call to ReadServiceGroups():");
    Console.WriteLine(ex.GetErrorMessages());
}
```

3.2 Operation «Lese Basisleistungen»

Mit der Operation **readBasicServices** können die momentan verfügbaren Basisleistungen mit entsprechenden Servicecodes jeder Dienstleistungsgruppe abgefragt werden.

```
== Java Code: ==

BarcodeWebService service = new BarcodeWebService(Language.DE);
service.setLoginCredentials("username", "passwort");

// First read all service groups with ServiceGroupID
List<ServiceGroup> groups;
try {
    groups = service.readServiceGroups();
} catch (BarcodeErrorException e) {
    System.out.println("Error in readServiceGroups:");
    System.out.println(e.getErrorMessages());
    return;
}

// basic services per service group
System.out.println("Available basic service codes:");
System.out.println("==============================");
for (ServiceGroup group : groups) {
   System.out.println("Basic Services for '" + group.getDescription() +
                                                          "':");
   try {
        List<BasicService> basicServices =
                        service.readBasicServices(group.getServiceGroupID());
        for (BasicService basicService : basicServices) {
            System.out.println("  - " + concatStrings(basicService.getPRZL())
                                      + ": " + basicService.getDescription());
        }
    } catch (BarcodeErrorException e) {
        // usually means that this service group is not available yet
        System.out.println(e.getErrorMessages());
    }
}
```

== C# Code: ==

```
BarcodeWebService service = new BarcodeWebService(Language.de);
service.UserName = "username";
service.Password = "password";

// First read Service-Groups with ServiceGroupID
List<PostLogistics.WebServiceBarcode.Client.ServiceGroup> groups;
try
{
    groups = service.ReadServiceGroups();
}
catch (BarcodeErrorException ex)
{
    Console.WriteLine("Error in ReadServiceGroups():");
    Console.WriteLine(ex.GetErrorMessages());
}
Console.WriteLine("Available basic service codes:");
Console.WriteLine("------------------------------");
foreach (PostLogistics.WebServiceBarcode.Client.ServiceGroup group in groups)
{
    // basic services per service group
    Console.WriteLine("Basic Services for {0}:", group.Description);
    try
    {
        List<PostLogistics.WebServiceBarcode.Client.BasicService> basicServices =
            service.ReadBasicServices(group.ServiceGroupID);
        foreach (PostLogistics.WebServiceBarcode.Client.BasicService basicService in
basicServices)
        {
            Console.WriteLine(" - {0}", basicService.PRZL);
        }
    }
    catch (BarcodeErrorException ex)
    {
        // usually means that this service group is not available yet
        Console.WriteLine(ex.GetErrorMessages());
    }
}
```

3.3 Operation «Lese Basisleistungen einer Frankierlizenz»

Mit der Operation **readAllowedServicesByFrankingLicense** können die für eine bestimmte Frankierlizenz verfügbaren Dienstleistungsgruppen und Basisleistungen abgefragt werden.

```
== Java Code: ==

BarcodeWebService service = new BarcodeWebService(Language.DE);
service.setLoginCredentials("username", "passwort");

// Read the allowed services for a given franking license
String frankingLicense = "franking license";
try {
    List<ReadAllowedServicesByFrankingLicenseResponse.ServiceGroups> allowedServices =
service.readAllowedServicesByFrankingLicense(frankingLicense, Language.DE);
    System.out.println("Allowed services for the franking license 60022220:");
    for (ReadAllowedServicesByFrankingLicenseResponse.ServiceGroups sgs :
allowedServices) {
        ServiceGroup sg = sgs.getServiceGroup();
        System.out.println("ServiceGroupDescription: " + sg.getDescription());
        System.out.println("PostID: " + sg.getServiceGroupID());
        System.out.println("Associated BasicServices with Description and PRZLs");
        for (BasicService bs : sgs.getBasicService()) {
            System.out.println("BasicServiceDescription: "+bs.getDescription());
            System.out.println("Associated PRZLs: ");
            int counter = 1;
            for (String s : bs.getPRZL()) {
                System.out.println("PRZL " + counter + ": " + s);
                counter++;
            }
        }
    }
} catch (BarcodeErrorException e) {
    // this occurs if the basic service does not exist.
    System.out.println("Errors: ");
    System.out.println(e.getErrorMessages());
}
```

== C# Code: ==

```
BarcodeWebService service = new BarcodeWebService(Language.de);
service.UserName = "username";
service.Password = "password";

// Read allowedServices with a test franking license
List<AllowedService> allowedServices;
int listCounter = 1;
try
{
    allowedServices = service.ReadAllowedServicesByFrankingLicense(FrankingLicense);
    foreach (AllowedService allowedService in allowedServices)
    {
        PostLogistics.WebServiceBarcode.Client.ServiceGroup sg =
allowedService.ServiceGroup;
        Console.WriteLine("********** ServiceGroups Element Nr. " + listCounter +
" **********");
        Console.WriteLine("ServiceGroupDescription: " + sg.Description);
        Console.WriteLine("PostID: " + sg.ServiceGroupID);
        Console.WriteLine("********** List of BasicServices **********");
        foreach (PostLogistics.WebServiceBarcode.Client.BasicService bs in
allowedService.BasicServices)
        {
            Console.WriteLine("********** BasicService Element **********");
            Console.WriteLine("BasicServiceDescription: " + bs.Description);
            int counter = 1;
            if (bs.PRZL != null)
            {
                foreach (String przl in bs.PRZL)
                {
                    Console.WriteLine("PRZL " + counter + ": " + przl);
                    counter++;
                }
            }
        }
        Console.WriteLine("");
        Console.WriteLine("********** (END ServiceGroups Element Nr. " + listCounter +
") **********");
        Console.WriteLine("");
        listCounter++;
    }
}
catch (BarcodeErrorException ex)
{
    Console.WriteLine(Fehler in WebService Aufruf
ReadAllowedServicesByFrankingLicense():");
    Console.WriteLine(ex.GetErrorMessages());
}
```

3.4 Operation «Lese Zusatzleistungen»

Mit der Operation **readAdditionalServices** können die momentan verfügbaren Zusatzleistungen mit entsprechenden Servicecodes zu einer Basisleistung abgefragt werden.

```
== Java Code: ==

BarcodeWebService service = new BarcodeWebService(Language.DE);
service.setLoginCredentials("username", "passwort");

// Read available additional service codes for "Sperrgut Priority".
try {
    List<AdditionalService> additionalServices = service.readAdditionalServices
(BasicServiceCode.PRI, BasicServiceCode.SP);
    System.out.println("Available additonal services for PRI, SP:");
    for (AdditionalService additionalService : additionalServices) {
        System.out.println(additionalService.getPRZL() + ": " +
        additionalService.getDescription());
    }
} catch (BarcodeErrorException e) {
    // this occurs if the basic service does not exist
System.out.println("Error in readAdditionalServices(): " + e.getErrorMessages());
}
```

```
== C# Code: ==

BarcodeWebService service = new BarcodeWebService(Language.de);
service.UserName = "username";
service.Password = "password";

// Read available additional service codes for "Sperrgut Priority".
try
{
    List<AdditionalService> additionalServices =
        service.ReadAdditionalServices(BasicServiceCode.PRI,
        BasicServiceCode.SP);
    Console.WriteLine("Available additional services for PRI, SP:");
    foreach (AdditionalService additionalService in additionalServices)
    {
        Console.WriteLine("{0} : {1}", additionalService.PRZL,
           additionalService.Description);
    }
}
catch (BarcodeErrorException ex)
{
    // this occurs if the basic service does not exist.
    Console.WriteLine("Error in ReadAdditionalServices():");
    Console.WriteLine(ex.GetErrorMessages());
}
```

3.5 Operation «Lese Zustellanweisungen»

Mit der Operation **readDeliveryInstructions** können die momentan verfügbaren Zustellanweisungen mit entsprechenden Servicecodes zu einer Basisleistung abgefragt werden.

```
== Java Code: ==

BarcodeWebService service = new BarcodeWebService(Language.DE);
service.setLoginCredentials("username", "passwort");

// Read available delivery instruction codes for basic service PRI, SP:
try {
    List<DeliveryInstructions> deliveryInstructions =
        service.readDeliveryInstructions(BasicServiceCode.PRI,
            BasicServiceCode.SP);
    System.out.println("Available delivery instructions for
        basic service PRI, SP:");
    for (DeliveryInstructions zaw : deliveryInstructions) {
        System.out.println(zaw.getPRZL() + ": "  + zaw.getDescription());
    }
} catch (BarcodeErrorException e) {
    // this occurs if the basic service does not exist.
    System.out.println("Errors: ");
    System.out.println(e.getErrorMessages());
}
```

== C# Code: ==

```
BarcodeWebService service = new BarcodeWebService(Language.de);
service.UserName = "username";
service.Password = "password";

// First get all available service groups.
List<PostLogistics.WebServiceBarcode.Client.ServiceGroup> groups =
service.ReadServiceGroups();
// Get available label layouts for each service group and display them
Console.WriteLine ("Supported delivery instructions per basic service ...");
Console.WriteLine ("=====================================================");
foreach (PostLogistics.WebServiceBarcode.Client.ServiceGroup group in groups)
{
    Console.WriteLine("for ServiceGroup '" + group.Description + "':");
    try {
        List<PostLogistics.WebServiceBarcode.Client.BasicService> basicServices =
service.ReadBasicServices(group.ServiceGroupID);
        foreach (PostLogistics.WebServiceBarcode.Client.BasicService basicService in
basicServices)
        {
             Console.WriteLine("  - " + basicService.PRZL + ": ");
             String[] basicServiceCodes = basicService.PRZL;
             List<DeliveryInstruction> deliveryInstructions =
service.ReadDeliveryInstructions(basicServiceCodes);
            foreach (DeliveryInstruction deliveryInstruction in deliveryInstructions)
            {
                Console.WriteLine(deliveryInstruction.PRZL);
                Console.WriteLine(", ");
            }
            Console.WriteLine ("");
            }
        }
        catch (BarcodeErrorException ex)
        {
        // This usually means, that this service group is currently not supported at all.
        Console.WriteLine("Error in WebService call to ReadDeliveryInstructions:");
        Console.WriteLine(ex.GetErrorMessages());
    }
}
```

3.6 Operation «Lese Darstellungsarten»

Mit der Operation **readLabelLayouts** können die momentan verfügbaren Darstellungsarten für eine gültige Kombination von PRZLs abgefragt werden.

== Java Code: ==

```
BarcodeWebService service = new BarcodeWebService(Language.DE);
service.setLoginCredentials("username", "passwort");

// First get all available service groups with serviceGroupID.
List<ServiceGroup> groups;
try {
    groups = service.readServiceGroups();
} catch (BarcodeErrorException e) {
    System.out.println("Error in readServiceGroups(): ");
    System.out.println(e.getErrorMessages());
    return;
}

// Get available label layouts for each service group and display them
System.out.println("Available Label Layouts:");
System.out.println("========================");
for (ServiceGroup group : groups) {
    System.out.println("LabelLayouts for '" + group.getDescription() + "':");
    List<BasicService> basicServices;
   try {
      basicServices = service.readBasicServices(group.getServiceGroupID());
   } catch (BarcodeErrorException e) {
        System.out.println("Error in readBasicServices(): ");
        System.out.println(e.getErrorMessages());
        return;
}
for (BasicService basicService : basicServices) {
   try {
       System.out.println("for BasicService " + basicService.getDescription());
       // read label layouts with a valid combination of PRZLs (here extracted from a
given BasicService)
       List<LabelLayoutResponse> layouts = service.readLabelLayouts
(basicService.getPRZL());
       for (LabelLayoutResponse layout : layouts) {
            System.out.print(" - " + layout.getLabelLayout() + ": ");
            System.out.print(" max. " + layout.getMaxServices() + " services ");
            System.out.print(" and " + layout.getMaxDeliveryInstructions()
             + " delivery instructions per label, ");
            if (layout.isFreeTextAllowed()) {
               System.out.println("freetext allowed.");
            } else {
                System.out.println("freetext not allowed.");
            }
        }
   } catch (BarcodeErrorException e) {
       // This usually means, that this service group is currently not supported at all.
       System.out.println(e.getErrorMessages());
   }
}
```

== C# Code: ==

```
BarcodeWebService service = new BarcodeWebService(Language.de);
service.UserName = "username";
service.Password = "password";

// First get all available service groups with serviceGroupID.
List<PostLogistics.WebServicebarcode.Client.ServiceGroup> groups;
try
{
    groups = service.ReadServiceGroups();
}
catch (BarcodeErrorException ex)
{
    Console.WriteLine("Error in ReadServiceGroups():");
    Console.WriteLine(ex.GetErrorMessages());
}
// Get available label layouts for each service group and display them
Console.WriteLine("Available Label Layouts:");
Console.WriteLine("========================");
foreach (PostLogistics.WebServicebarcode.Client.ServiceGroup group in groups)
{
    Console.WriteLine(LabelLayouts for '{0}':", group.Description);
    List<PostLogistics.WebServiceBarcode.Client.BasicService> basicServices;
    try
    {
        basicServices = service.ReadBasicServices(group.ServiceGroupID);
        foreach (PostLogistics.WebServiceBarcode.Client.BasicService basicService in
basicServices)
        {
            try
            {
            Console.WriteLine("for BasicService " + basicService.Description);
            // read label layouts with a valid combination of PRZLs (here extracted from
a given BasicService)
            List<PostLogistics.WebServiceBarcode.Client.LabelLayout> layouts =
service.ReadLabelLayouts(basicService.PRZL);
            Foreach (PostLogistics.WebServiceBarcode.Client.LabelLayout layout in
layouts)
            {
                Console.WriteLine(" - " + layout.Layout + ": ");
                Console.WriteLine(" max. " + layout.MaxServices + " services ");
                Console.WriteLine(" and " + layout.MaxDeliveryInstructions + " delivery
instructions per label, ");
                if (layout.FreeTextAllowed)
                {
                    Console.WriteLine("freetext allowed.");
                }
                else
                {
```

```
                    Console.WriteLine("freetext not allowed.");
                }
            }
        }
        catch (BarcodeErrorException ex)
        {
            Console.WriteLine(ex.GetErrorMessages());
        }
    }
}
catch (BarcodeErrorException ex)
{
    Console.WriteLine("Error in ReadBasicServices(): ");
    Console.WriteLine(ex.GetErrorMessages());
}
```

4 Validierungsoperation

Der Webservice stellt eine Operation zur Verfügung, um die Kombination der Dienstleistungscodes eines Label-Requests zu validieren. Es wird überprüft, ob die im Request enthaltenen Servicecodes (Basisleistung, Zusatzleistungen und Zustellanweisungen) miteinander kombiniert werden dürfen. Ebenfalls wird überprüft, ob die verwendeten Codes im gewählten Labelformat (A5, A6, A7 oder FE) erlaubt sind. Je nach Format ist nur eine gewisse Anzahl an Dienstleistungscodes und Zustellanweisungen erlaubt.

Die Schnittstelle der Client API Libraries verwenden als Input für die Validierung direkt die gleichen Barcode-Label-Request-Klassen, wie die **generateLabel-**Methoden. Dies hat den Vorteil, dass die erstellten Requests vorgängig einfach auf korrekte Dienstleistungscode-Kombinationen überprüft werden können. Die gleiche Validierung wird selbstverständlich auch bei einem Aufruf der **generateLabel-**Methoden vom Webservice vorgenommen.

Die aktuell verfügbaren Dienstleistungscodes stehen in speziellen Konstantenklassen als String-Konstanten zur Verfügung: **BasicServiceCode, AdditionalServiceCode** und **DeliveryInstructionCode.** Diese Klassen enthalten nur die Codes, die zum Zeitpunkt der Entwicklung des Webservice-Barcode-Client-API verfügbar waren. Weitere Codes können unter Umständen später hinzukommen und als normale Strings übergeben werden. Um alle aktuellen Codes zu erhalten, sind die Lese-Operationen zu verwenden.

Das folgende Beispiel demonstriert die Überprüfung einer Kombination von Basisleistungscodes, Zusatzleistungscodes und Zustellanweisungscodes.

== Java Code: ==

```
BarcodeWebService service = new BarcodeWebService(Language.DE);
service.setLoginCredentials("username", "passwort");

// Set label layout to validate allowed service codes for
// (otherwise the default is used)
service.setLabelLayout(LabelFormat.A5);

// Label request with basic service code(s) (PRI = Priority)
BarcodeLabelRequest request = service.createLabelRequest();
request.addServiceCode(BasicServiceCode.PRI);

// additional service codes
request.addServiceCode(AdditionalServiceCode.FRA);
request.addServiceCode("SI");

// delivery instruction codes
request.addServiceCode(DeliveryInstructionCode.ZAW3213);
request.addServiceCode("ZAW3215");

// set the country required to validate the combination
request.setCountryIsoCode("CH");

// validation
ValidationResponse result = service.validateCombination(request);
if (!result.hasErrors()) {
    System.out.println("Validation was successful.");
} else {
    System.out.println("Validation has ERRORS: ");
    for (BarcodeError error : result.getErrors()) {
        System.out.println(error.getCode() + " - " + error.getMessage());
    }
}
if (result.hasWarnings()) {
    System.out.println("WARNINGS: ");
    for (BarcodeWarning warning : result.getWarnings()) {
        System.out.println(warning.getCode() + " - " + warning.getMessage());
    }
}
```

== C# Code: ==

```
BarcodeWebService service = new BarcodeWebService(Language.de);
service.UserName = "username";
service.Password = "password";

// Set label layout to validate allowed service codes for
// (otherwise the default is used)
service.LabelLayout = LabelFormat.A5;

// Label request with basic service code(s) (PRI = Priority)
BarcodeLabelRequest request = service.CreateLabelRequest();
request.AddServiceCode(BasicServiceCode.PRI);

// additional service codes
request.AddServiceCode(AdditionalServiceCode.FRA);
request.AddServiceCode("SI");

// delivery instruction codes
request.AddServiceCode(DeliveryInstructionCode.ZAW3213);
request.AddServiceCode("ZAW3215");

// set the country required to validate the combination
request.AddCountryIsoCode("CH");

// validation
ValidationResponse result = service.ValidateCombination(request);
if (!result.HasErrors())
{
    Console.WriteLine("Validation was successful.");
}
else
{
    Console.WriteLine("Validation has ERRORS: ");
    foreach (ErrorsTypeError error in result.GetErrors())
    {
        Console.WriteLine(error.Code + " - " + error.Message);
    }
}
if (result.HasWarnings())
{
    Console.WriteLine("Validation has WARNINGS: ");
    foreach (WarningsTypeWarning warning in result.GetWarnings())
    {
        Console.WriteLine(warning.Code + " - " + warning.Message);
    }
}
```

Dieses Beispiel validiert ohne Fehler.

Wird hingegen in den Beispielen «ZAW3215» durch «ZAW3222» ersetzt, so liefert der Webservice Aufruf entsprechende Validierungs-Fehlermeldungen zurück.

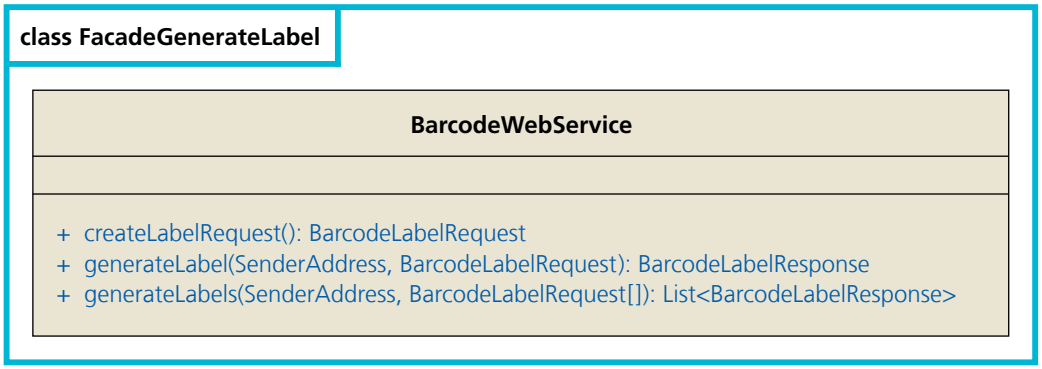
Es gibt noch eine weitere Methode, nämlich **validateCombinations,** mit der gleich mehrere **BarcodeLabelRequests** in einem Webserviceaufruf validiert werden können.

5 Label / Barcode Generierung

5.1 Label Generierung

Die Operation zur Generierung von Adressträgern erwartet als Input zwei Parameter:
- **SenderAddress:** die Absenderadresse
- **LabelRequest:** die gewünschten Parameter inklusive Empfängeradresse für einen zu erstellenden Adressträger

Es stehen zwei Methoden zum Aufruf dieser Operation zur Verfügung: eine zum Erstellen eines einzigen Adressträgers und eine zum Erstellen mehrerer Adressträger mit Barcode in einem Aufruf.



Um eine **BarcodeLabelRequest-**Instanz zu erzeugen, ist die Factory-Methode **createLabelRequest** zu verwenden. Diese Methode stellt sicher, dass die Requests bereits mit einer fortlaufenden **itemID** initialisiert werden. Die **itemID** kann bei Bedarf auch selbstständig gesetzt werden. In diesem Falle ist allerdings die Eindeutigkeit der **itemIDs** innerhalb eines **generateLabels-**Aufrufs sicherzustellen.

Die generierten Labels werden als BarcodeLabelResponse-Objekte zurückgegeben. Eine BarcodeLabelResponse kann wiederum Fehlermeldungen (in der gewählten Sprache) enthalten, falls die Generierung des entsprechenden Adressträgers mit den gewünschten Attributen nicht möglich war. Ansonsten ist der generierte Adressträger im entsprechenden File-Format in der Property **LabelBinaryData** enthalten. Diese binären Daten können zum Beispiel als Bilddatei abgespeichert werden.

Das folgende Beispiel demonstriert, wie ein einfacher, einzelner Adressträger generiert und die Response ausgelesen werden kann.

== Java Code: ==

```
BarcodeWebService service = new BarcodeWebService(Language.DE);
service.setLoginCredentials("username", "passwort");

// Franking licence (mandatory!):
// this has to be a valid license for your barcode web service user account
service.setFrankingLicense("12345678");


// enable/disable print preview
service.setPrintPreview(false);

// sender address
SenderAddress sender = new SenderAddress();
sender.setName1("Meier AG");
sender.setStreet("Viktoriaplatz 10");
sender.setZIP("8048");
sender.setCity("Zürich");

// BarcodeLabelRequest with recipient address
BarcodeLabelRequest labelRequest = service.createLabelRequest();
RecipientAddress address = labelRequest.getAddress();
address.setFirstName("Melanie");
address.setName1("Steiner");
address.setName2 ("Müller AG");
address.setName3 ("Abteilung Marketing");
address.setAddressSuffix ("Gebäude 6-Ost");
address.setStreet("Viktoriastrasse");
address.setHouseNo("21a");
address.setZIP("3030");
address.setCity("Bern");

// Add notification services with Builder-pattern and immediate validation
NotificationService.Builder notificationServiceBuilder =
   new NotificationService.Builder(
      NotificationService.NotificationType.CODE_1, Language.DE);
NotificationService notificationService =
   notificationServiceBuilder.email("test@test.ch")
                             .freeText1("Test 1")
                             .freeText2("Test 2").build();
labelRequest.addNotificationService(notificationService);

// set service-code
labelRequest.addServiceCode(BasicServiceCode.ECO);

// Service call generateLabel:
BarcodeLabelResponse response = service.generateLabel(sender, labelRequest);

// display response information
try {

    // read binary label image data
    // (throws an Exception if error in the response)
    byte[] binaryData = response.getLabelBinaryData();
```

```
      // dsplay barcode information
    System.out.println("Barcode successfully generated: " +
          response.getIdentCode());
    System.out.print(" Binary-Data: " + response.getImageFileType());
    System.out.print(", " + response.getImageResolution() + " DPI, ");
    System.out.print(response.isColorPrintRequested() ? "color, " : "black, ");
    System.out.println(binaryData.length + " bytes");

} catch (BarcodeErrorException e) {

// label not generated, display errors:
// (error messages are in the choosen language)
System.out.println("Barcode was not generated due to errors: ");
    System.out.println(e.getErrorMessages());
}

// Also display warning messages
if (response.hasWarnings()) {
    System.out.print(", WARNINGS:");
    System.out.println(concatMessages(response.getWarnings()));
}
```

== C# Code: ==

```
BarcodeWebService service = new BarcodeWebService(Language.de);
service.UserName = "username";
service.Password = "password";

// Franking licence (mandatory!):
// this has to be a valid license for your barcode web service user account
service.FrankingLicense = "12345678";
service.PrintPreview = false;

BarcodeLabelRequest labelRequest = service.CreateLabelRequest();

// Sender address
SenderAddress sender = new SenderAddress();
sender.Name1 = "Meier AG";
sender.Street = "Viktoriaplatz 10";
sender.Zip = "8048";
sender.City = "Zürich";

// BarcodeLabelRequest with recipient address
RecipientAddress address = labelRequest.Address;
address.Title = "Frau";
address.FirstName = "Melanie";
address.Name1 = "Steiner";
address.Name2 = "Müller AG";
address.Name3 = "Abteilung Marketing";
address.AddressSuffix = "Gebäude 6-Ost";
address.Street = "Viktoriastrasse";
address.HouseNo = "21a";
address.Zip = "3030";
address.City = "Bern";
```

```
// Add notification services
NotificationService.Builder notificationServiceBuilder = new NotificationService.Builder(
    NotificationService.NotificationType.CODE_1, Language.de);
NotificationService notificationService = notificationServiceBuilder.email
("test@test.ch").freeText1("Test 1").
    freeText2("Test 2").build();
labelRequest.AddNotificationService(notificationService);

// set service-code
labelRequest.AddServiceCode(BasicServiceCode.ECO);

// Service call generateLabel:
BarcodeLabelResponse response = service.GenerateLabel(sender, labelRequest);

// display response information
try
{
    // Read binary label image data
    // (throws an Exception if error in the response)
    byte[] binaryData = response.LabelBinaryData;

    // display barcode information
    Console.Write("Barcode successfully generated: " + response.IdentCode);
    Console.Write(" Binary-Data: " + response.ImageFileType);
    Console.Write(", " + response.ImageResolution + " DPI, ");
    Console.Write(response.ColorPrintRequested ? "color, " : "black, ");
    Console.WriteLine(binaryData.Length + " bytes");
}
catch (BarcodeErrorException ex)
{
    // label not generated, display errors:
     // (error messages are in the choosen language)
    Console.WriteLine("Barcode was not generated due to errors: ");
    Console.WriteLine(ex.GetErrorMessages());
}

// Also display warning messages
if (response.HasWarnings())
{
    Console.Write(", WARNINGS:");
    Console.WriteLine(response.GetWarnings());
}
```

5.2 Einzelbarcode Generierung

Mit der Operation **generateSingleBarcodes** können die Barcodes eines Adressträgers einzeln bezogen werden. Sie ähnelt sehr stark der Operation **generateLabel** und unterscheidet sich lediglich darin, dass im Aufruf gewisse Parameter(PrintAdresses, Printpreview, labelLayout) nicht gesetzt werden können. Auch die Antwort unterscheidet sich nur minim. Anstelle eines einzelnen Labels wird eine Liste von Barcodes retourniert.

```
== Java Code: ==

BarcodeWebService service = new BarcodeWebService(Language.DE);
service.setLoginCredentials("username", "passwort");

// Franking licence (mandatory!):
// this has to be a valid license for your barcode web service user account
service.setFrankingLicense("12345678");

// enable/disable print preview
service.setPrintPreview(false);
// sender address
SenderAddress sender = new SenderAddress();
sender.setName1("Meier AG");
sender.setStreet("Viktoriaplatz 10");
sender.setZIP("8048");
sender.setCity("Zürich");
sender.setDomicilePostOffice("8048 Zürich");

// BarcodeLabelRequest with recipient address
SingleBarcodesRequest labelRequest = service.createSingleBarcodesRequest();
RecipientAddress address = labelRequest.getAddress();
address.setTitle("Frau");
address.setFirstName("Melanie");
address.setName1("Steiner");
address.setStreet("Viktoriastrasse");
address.setHouseNo("21a");
address.setZIP("3030");
address.setCity("Bern");

// set service-code
labelRequest.addServiceCode("RINL");
labelRequest.addServiceCode("eAR");
labelRequest.addServiceCode("ZAW2511");

// Service call generateLabel:
SingleBarcodesResponse response = service.generateSingleBarcodes(sender, labelRequest);

// display response information
try {
    // Read binary barcodes data
    // (throws an Exception if error in the response)
    List<byte[]> binaryData = response.getBarcodesBinaryData();
```

```
    // Informationen zum Bar-Code ausgeben
    System.out.println("Barcode successfully generated: " + response.getIdentCode());
    System.out.print(" Binary-Data: " + response.getImageFileType());
    System.out.print(", " + response.getImageResolution() + " DPI, ");
    System.out.print(response.isColorPrintRequested() ? "color, " : "black, ");
    System.out.println("Number of generated single barcodes: " + binaryData.size());
    for (int i = 0; i < binaryData.size(); i++) {
        System.out.println("Barcode " + i + "  " + binaryData.get(i).length + " bytes");
    }
} catch (BarcodeErrorException e) {
    // label not generated, display errors:
    // (error messages are in the chosen language)
    System.out.println("Barcode was not generated due to errors: ");
    System.out.println(e.getErrorMessages());
}

// also display warning messages
if (response.hasWarnings()) {
    System.out.println(" with WARNINGS:");
    for (BarcodeWarning warning : response.getWarnings()) {
        System.out.println("  " + warning.getCode() + " - " + warning.getMessage() + ", ");
}
```

== C# Code: ==

```
BarcodeWebService service = new BarcodeWebService(Language.de);
service.UserName = "username";
service.Password = "password";
service.FrankingLicense = FrankingLicense;
service.PrintPreview = false;

// Sender address
SenderAddress sender = new SenderAddress();
sender.Name1 = "Meier AG";
sender.Street = "Viktoriaplatz 10";
sender.Zip = 8048L;
sender.City = "Zürich";
sender.DomicilePostOffice = "Zürich";

// BarcodeLabelRequest inklusive recipient address
SingleBarcodesRequest request = service.CreateSingleBarcodesRequest();
RecipientAddress address = request.Address;
address.setTitle("Frau");
address.setFirstName("Melanie");
address.setName1("Steiner");
address.setStreet("Viktoriastrasse");
address.setHouseNo("21a");
address.setZIP("3030");
address.setCity("Bern");

request.AddServiceCode("RINL");
request.AddServiceCode("eAR");
request.AddServiceCode("ZAW2511");

// call the service to create the single barcodes
SingleBarcodesResponse response = service.GenerateSingleBarcodes(sender, request);

try
{
    // Read binary label image data
    // (throws an Exception if error in the response)
    byte[] binaryData = response.LabelBinaryData;
}
catch (BarcodeErrorException ex)
{
    // label not generated, display errors:
    // (error messages are in the choosen language)
    Console.WriteLine("Barcode was not generated due to errors: ");
    Console.WriteLine(ex.GetErrorMessages());
}

// Also display warning messages
if (response.HasWarnings())
{
    Console.Write(", WARNINGS:");
    Console.WriteLine(response.GetWarnings());
}
```

5.3 Operation «Generiere Barcode»

Mit der Operation **generateBarcode** kann ein Kunde vordefinierte Barcodes generieren und ausdrucken.

```
== Java Code: ==

BarcodeWebService service = new BarcodeWebService(Language.DE);
service.setLoginCredentials("username", "passwort");

// Generate an LSO-1 barcode

// create first the request object for the generateBarcode request
IndividualBarcodeRequest request = service.createIndividualBarcodeRequest();
// choose which barcode to generate (here LSO-1)
BarcodeDefinition barcodeDefinition = new BarcodeDefinition();
barcodeDefinition.setImageFileType(ImageFileType.PDF);
barcodeDefinition.setImageResolution(300);
barcodeDefinition.setBarcodeType(BarcodeType.LSO_1);
request.setBarcodeDefinition(barcodeDefinition);

// service call generateBarcode:
IndividualBarcodeResponse response = service.generateBarcode(request);

try {
    // read binary barcode data
    // (throws an Exception if there are errors in the response)
    byte[] binaryData = response.getBarcodeData();

    // display barcode information
    System.out.println("Barcode successfully generated!");
    System.out.print(" Binary-Data: " + response.getBarcodeData());
    System.out.print(", " + response.getBarcodeDefinition().getImageResolution() +
" DPI, ");
    System.out.print(response.isColorPrintRequired() ? "color, " : "black, ");
    System.out.println(binaryData.length + " bytes");

} catch (BarcodeErrorException e) {
    // barcode not generated, display errors:
    // (error messages are in the chosen language)
    System.out.println("Barcode was not generated due to errors: ");
    System.out.println(e.getErrorMessages());
}
```

```
== C# Code: ==

BarcodeWebService service = new BarcodeWebService(Lang);

service.UserName = UserName;
service.Password = Password;
service.EndpointAddress = new EndpointAddress(Url);

IndividualBarcodeRequest request = service.CreateIndividualBarcodeRequest();
BarcodeDefinition barcodeDefinition = new BarcodeDefinition();
barcodeDefinition.BarcodeType = BarcodeType.LSO_1;
barcodeDefinition.ImageFileType = ImageFileType.JPG;
barcodeDefinition.ImageResolution = 300;
request.Definition = barcodeDefinition;

IndividualBarcodeResponse response = service.GenerateBarcode(request);
try
{
    // Read binary label image data
    // (throws an Exception if error in the response)
    byte[] binaryData = response.BarcodeData;
}
catch (BarcodeErrorException ex)
{
    // barcode not generated, display errors:
    // (error messages are in the choosen language)
    Console.WriteLine("Barcode was not generated due to errors: ");
    Console.WriteLine(ex.GetErrorMessages());
}
if (response.HasWarnings())
{
    Console.Write(", WARNINGS:");
    Console.WriteLine(response.GetWarnings());
}
```

5.4 Beispiele für Beleglose Nachnahme (BLN)

Für die beleglose Nachnahme müssen nebst dem ServiceCode BLN auch diverse Felder erfasst werden. Diese unterscheiden sich für den Fall BLN mit ESR und BLN mit IBAN.

Achtung: BLN mit IBAN darf aktuell noch nicht verwendet werden.

BLN mit ESR:

```
== Java Code: ==

labelRequest.addServiceCode(AdditionalServiceCode.BLN);

labelRequest.setCashOnDeliveryAmount(10f);
labelRequest.setCashOnDeliveryRefNrESR(„965993000000000000001237460“);
labelRequest.setCashOnDeliveryCustomerNrESR(„010003757“);
labelRequest.setCashOnDeliveryCustomerEmail(„hans.muster@mail.ch“);
labelRequest.setCashOnDeliveryCustomerMobile(„0791234567“);
```

```
== C# Code: ==

labelRequest.AddServiceCode(AdditionalServiceCode.BLN);

labelRequest.CashOnDeliveryAmount = 10;
labelRequest.CashOnDeliveryRefNrESR = „965993000000000000001237460“;
labelRequest.CashOnDeliveryCustomerNrESR = „010003757“;
labelRequest.CashOnDeliveryCustomerEmail = „hans.muster@mail.ch“;
labelRequest.CashOnDeliveryCustomerMobile = „0791234567“;
```

BLN mit IBAN:

```
== Java Code: ==

labelRequest.addServiceCode(AdditionalServiceCode.BLN);

labelRequest.setCashOnDeliveryAmount(10f);
labelRequest.setCashOnDeliveryIbanNumber(„CH10002300A1023502601“);
labelRequest.setCashOnDeliveryIbanName(„Hans Muster“);
labelRequest.setCashOnDeliveryIbanStreet(„Musterstrasse 11“);
labelRequest.setCashOnDeliveryIbanZip(„3011“);
labelRequest.setCashOnDeliveryIbanCity(„Bern“);
labelRequest.setCashOnDeliveryCustomerEmail(„hans.muster@mail.ch“);
labelRequest.setCashOnDeliveryCustomerMobile(„0791234567“);
```

```
== C# Code: ==

labelRequest.AddServiceCode(AdditionalServiceCode.BLN);

labelRequest.CashOnDeliveryAmount = 10;
labelRequest.CashOnDeliveryIbanNumber = „CH10002300A1023502601“;
labelRequest.CashOnDeliveryIbanName = „Hans Muster“;
labelRequest.CashOnDeliveryIbanStreet = „Musterstrasse 11“;
labelRequest.CashOnDeliveryIbanZip = „3011“;
labelRequest.CashOnDeliveryIbanCity = „Bern“;
labelRequest.CashOnDeliveryCustomerEmail = „hans.muster@mail.ch“;
labelRequest.CashOnDeliveryCustomerMobile = „0791234567“;
```

Post CH AG
Support Webservices
Viktoriastrasse 21
Postfach
3030 Bern

webservice@post.ch
www.post.ch/webservice

DIE POST

11.2013