# BİLGE NCTS Trader Portal Servis Tanımları

Bu döküman yeni Bilge Projesi NCTS Faz-5 kapsamında sağlanan trader portal servislerinin nasıl kullanılacağını anlatmaktadır.

Yeni servisler RESTful API olarak geliştirilmiştir. İlgili servislerle ilgili detaylar (servis imzaları, dönüş değerleri, hata kodları) dokümanın ilgili kısımlarında verilecektir.

Dokümanda kullanılan kod örnekleri Java yazılım dilinde yazılmıştır.

Trader portal servisleri için ana URL : ***“https://ws.gtb.gov.tr:8443/NCTSTraderApi/ “***

## Basic Authentication

NCTS Faz-5 web servisleri kurum Ticaret Bakanlığı tarafından sağlanan “Api Gateway”’in arkasında çalışmaktadır. Bu kapsamda, servis çağrımları sırasında Ticaret Bakanlığı tarafından firmanıza sağlanacak / sağlanmış olan “Basic Authentication” bilgilerinin kullanılması gerekmektedir.

## Kimlik Doğrulama

Servisleri kullanmaya başlamadan önce ilk olarak kimlik doğrulaması yapılması gerekmektedir. Bu kapsamda ilk olarak kullanıcılara sağlanan “kullanıcı kodu” ve “şifre” ile “login” servisi çağrılarak kimlik doğrulaması yapılması gerekmektedir. Bu servis doğru parametrelerle çağrıldığında bir token üretecektir. Üretilen bu token’ın her servis çağrımında parametre olarak geçilmesi gerekmektedir.

Sistemdeki tokenlerin geçerlilik süresi 15 dakikadır. Her servis çağrımında 15 dakikalık süre yenilenir. Token alındıktan sonra 15 dakika içinde herhangi bir servis çağrımı yapılmaz ise “kullanıcı kodu” ve “şifre” kullanılarak “login” servisi yeniden çağrılarak yeni bir token üretilmelidir.

“logout” servis kullanılarak ilgili session kapatımı yapılabilir. Bu servis çağrıldığında ilgili token geçersiz hale gelecektir.

### login(String jsonQuery)

***Servis Path : “/auth/login”***

Bu servis “kullanıcı kodu” ve “şifre” belirtilerek sisteme giriş yapılmasını sağlar.

**Servis İmzası:**

@POST  
@Path("/login")  
@Consumes(MediaType.*APPLICATION\_JSON*)  
@Produces(MediaType.*APPLICATION\_JSON*)  
public Response login(String jsonString)

**Servis Parametreleri:**

{

“userId” : “xxxxxxxxxx” (String),

“password”: “xxxxxxxxxxx” (String)

}

**Servis Dönüş Parametreleri (Başarılı) :**

{

"data" : {

"token" : "od9W8wYg7xR30BMaNicWrCRd6JM+3IQypakyYFKb+fTTU7YwIrK5jQ==",

"loginResult" : true

},

"error" : [ ],

"status" : 200

}

**Servis Dönüş Parametreleri (Başarısız) :**

{

"data" : null,

"error" : [ "User Id or password is incorrect" ],

"status" : 401

}

**Örnek Kullanımı:**

public void testLoginService() throws HTTPResponseException {  
 WSCommunicator wsCommunicator = new WSCommunicator("http://bilge.ncts.traderportal/auth/");  
 JsonObject payload;  
  
  
 // Test valid password  
 payload = new JsonObject();  
 payload.put(GeneralConstants.*AUTH\_SRV\_USER\_ID*, "xxxxxxxx");  
 payload.put(GeneralConstants.*AUTH\_SRV\_PASSWORD*, "xxxxxxx");  
 String result = wsCommunicator.postData(payload, "login");  
 JsonObject resultObject = new JsonObject(result);  
 System.*out*.println("\*\*\*\*\*\*\*\*\*\* Login Service Result \*\*\*\*\*\*\*\*\*\*");  
 System.*out*.println(resultObject.encodePrettily());  
}

### logout(String jsonQuery)

***Servis Path : “/auth/logout”***

Bu servis, “login”servisiyle oluşturulan token’ın geçersiz olmasını sağlar.

**Servis İmzası:**

@POST  
@Path("/logout")  
@Consumes(MediaType.*APPLICATION\_JSON*)  
@Produces(MediaType.*APPLICATION\_JSON*)  
public Response logout(String jsonString)

**Servis Parametreleri:**

{

“token” : “xxxxxxxxxxxxxxxxxxxxxxxxxx” (String)

}

**Servis Dönüş Parametreleri :**

{

"data" : {

"userSessionLogoutResult" : "Success"

},

"error" : [ ],

"status" : 200

}

**Örnek Kullanımı:**

public void testLogoutService() throws HTTPResponseException {  
 WSCommunicator wsCommunicator = new WSCommunicator("http://bilge.ncts.traderportal/auth/");  
 JsonObject payload;  
  
 // Check logout  
 payload = new JsonObject();  
 payload.put(GeneralConstants.*AUTH\_SRV\_TOKEN*, token);  
 result = wsCommunicator.postData(payload, "logout");  
 resultObject = new JsonObject(result);  
 System.*out*.println("\*\*\*\*\*\*\*\*\*\* Logout Result \*\*\*\*\*\*\*\*\*\*");  
 System.*out*.println(resultObject.encodePrettily());  
}

## Mesaj Gönderme

Bu servisler yeni bir CC015C mesajı gönderme, daha önce gönderilen mesajı CC013C mesajı ile değiştirme, gönderilen mesajlarla ilgili oluşan mesajları sorgulama amacıyla kullanılır.

### submitDeclaration(String jsonQuery)

***Servis Path : “/trader/submitDeclaration”***

Bu servis, yeni bir CC015C mesajı göndermek için kullanılır.

**Servis İmzası:**

@POST  
@Path("/submitDeclaration")  
@Consumes({MediaType.*APPLICATION\_JSON*})  
@Produces({MediaType.*APPLICATION\_JSON*})  
public Response submitDeclaration(String jsonQuery)

**Servis Parametreleri:**

**“msgContent”** e-imzalı olarak gönderiliyor ise “signFlag” true olarak gönderilmesi ve “msgContent”’in imzalanmış veriyi içermesi gerekmektedir. İmzalanacak olan “msgContent”’in içeriği JSON objesi olarak aşağıdaki şekilde olmalıdır:

{

“phase5XML” : “xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx” (String),

“nationXML”: “xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx” (String)

}

**İmzalanmış olarak gönderilen mesaj içeriği:**

{

“token”: “xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx” (String),

“firmId”: “xxxxxxxxxxxxxxxxx” (String),

“userId”: “xxxxxxxxxxxxxx” (String),

“msgType”: ”CC015C” (String),

“signFlag”: true (Boolean),

“msgContent”: “xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx” (String, imzalanmış içerik)

}

**İmzalanmamış olarak gönderilen mesaj içeriği:**

{

“token”: “xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx” (String),

“firmId”: “xxxxxxxxxxxxxxxxx” (String),

“userId”: “xxxxxxxxxxxxxx” (String),

“msgType”: ”CC015C” (String),

“signFlag”: false (Boolean),

“msgContent”: {

“phase5XML” : “xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx” (String),

“nationXML”: “xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx” (String)

}

}

**Not: İmzalanmamış gönderim sadece test amaçlı olarak kullanılacaktır. Sistem canlıya alındığında imzalanmamış içeriğin gönderimine izin verilmeyecektir.**

**Servis Dönüş Parametreleri (Başarılı) :**

{

"data" : {

"guid" : "859e99e2-25f9-4b64-ace2-2f5bb1f4d65f"

},

"error" : [ ],

"status" : 200

}

**Servis Dönüş Parametreleri (Başarısız) :**

{

"data" : null,

"error" : [ "Parameters are not valid. Please check" ],

"status" : 400

}

**Örnek Kullanımı:**

public void testSubmitDeclaration () throws Exception {  
 WSCommunicator wsCommunicator = new WSCommunicator("http://bilge.ncts.traderportal/trader/");  
 byte[] xmlBytes = Files.*readAllBytes*(Paths.*get*("C:\\NCTS\_XML\_FILES\\message.xml"));  
 String xmlMessage = new String(xmlBytes, StandardCharsets.*UTF\_8*);  
 xmlBytes = Files.*readAllBytes*(Paths.*get*("C:\\NCTS\_XML\_FILES\\nation.xml"));  
 String nationMessage = new String(xmlBytes, StandardCharsets.*UTF\_8*);  
 JsonObject payload = new JsonObject();  
 JsonObject msgContent = new JsonObject();  
 msgContent.put(GeneralConstants.*TRADER\_SRV\_PHASE5\_XML*, xmlMessage);  
 msgContent.put(GeneralConstants.*TRADER\_SRV\_NATION\_XML*, nationMessage);  
  
 payload.put(GeneralConstants.*TRADER\_SRV\_TOKEN*, getToken());  
 payload.put(GeneralConstants.*TRADER\_SRV\_FIRM\_ID*, "MilSOFT");  
 payload.put(GeneralConstants.*TRADER\_SRV\_USER\_ID*, "eserin");  
 payload.put(GeneralConstants.*TRADER\_SRV\_MSG\_TYPE*, "CC015C");  
 payload.put(GeneralConstants.*TRADER\_SRV\_SIGN\_FLAG*, false);  
 payload.put(GeneralConstants.*TRADER\_SRV\_MSG\_CONTENT*, msgContent);  
 String result = wsCommunicator.postData(payload, "submitDeclaration");  
 JsonObject resultObject = new JsonObject(result);  
 System.*out*.println(resultObject.encodePrettily());  
}

**CC0015-National Mesaj (nationXML) xsd:**

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

<xs:schema elementFormDefault="qualified"

xmlns:xs="http://www.w3.org/2001/XMLSchema">

<xs:element name="messageType" type="xs:string">

<xs:simpleType>

<xs:restriction base="xs:string">

<xs:pattern value="CC015National"/>

</xs:restriction>

</xs:simpleType>

</xs:element>

<xs:element name="taxStampPrice" type="xs:decimal" nillable="true">

<xs:simpleType>

<xs:restriction base="xs:decimal">

<xs:totalDigits value="4"/>

<xs:fractionDigits value="2"/>

</xs:restriction>

</xs:simpleType>

</xs:element>

<xs:element name="taxStampStatus" type="xs:string" nillable="true">

<xs:simpleType>

<xs:restriction base="xs:string"/>

</xs:simpleType>

</xs:element>

<xs:element name="taxStampDate" type="xs:string" nillable="true">

<xs:simpleType>

<xs:restriction base="xs:string"/>

</xs:simpleType>

</xs:element>

<xs:element name="exportDeclarations">

<xs:complexType>

<xs:sequence>

<xs:element type="xs:byte" name="declarationGoodsItemNumber"/>

<xs:element type="xs:string" name="exportDeclarationNo" minOccurs="0">

<xs:simpleType>

<xs:restriction base="xs:string">

<xs:length value="25"/>

</xs:restriction>

</xs:simpleType>

</xs:element>

<xs:element type="xs:string" name="exportDeclarationType">

<xs:simpleType>

<xs:restriction base="xs:string">

<xs:length value="9"/>

<xs:pattern value="[I]"/>

</xs:restriction>

</xs:simpleType>

</xs:element>

<xs:element type="xs:string" name="isPartial">

<xs:simpleType>

<xs:restriction base="xs:string">

<xs:length value="9"/>

<xs:pattern value="[YN]"/>

</xs:restriction>

</xs:simpleType>

</xs:element>

</xs:sequence>

</xs:complexType>

</xs:element>

<xs:element name="summarizedDeclarationOpenings">

<xs:complexType>

<xs:sequence>

<xs:element type="xs:string" name="openingType">

<xs:simpleType>

<xs:restriction base="xs:string">

<xs:length value="9"/>

<xs:pattern value="[23]"/>

</xs:restriction>

</xs:simpleType>

</xs:element>

<xs:element type="xs:string" name="summarizedDeclarationNo">

<xs:simpleType>

<xs:restriction base="xs:string">

<xs:length value="16"/>

</xs:restriction>

</xs:simpleType>

</xs:element>

<xs:element type="xs:string" name="atWareHouse">

<xs:simpleType>

<xs:restriction base="xs:string">

<xs:length value="9"/>

<xs:pattern value="[01]"/>

</xs:restriction>

</xs:simpleType>

</xs:element>

<xs:element type="xs:string" name="billOfLandingNo">

<xs:simpleType>

<xs:restriction base="xs:string">

<xs:length value="100"/>

</xs:restriction>

</xs:simpleType>

</xs:element>

<xs:element type="xs:string" name="wareHouseCode">

<xs:simpleType>

<xs:restriction base="xs:string">

<xs:length value="20"/>

</xs:restriction>

</xs:simpleType>

</xs:element>

<xs:element type="xs:string" name="billOfLandingLineNo">

<xs:simpleType>

<xs:restriction base="xs:string">

<xs:length value="6"/>

</xs:restriction>

</xs:simpleType>

</xs:element>

<xs:element type="xs:string" name="packageCount">

<xs:simpleType>

<xs:restriction base="xs:string">

<xs:length value="6"/>

</xs:restriction>

</xs:simpleType>

</xs:element>

<xs:element type="xs:string" name="isOpenedWithDifferentRegime"/>

</xs:sequence>

</xs:complexType>

</xs:element>

<xs:element name="wareHouseOpenings">

<xs:complexType>

<xs:sequence>

<xs:element type="xs:string" name="wareHouseDeclarationNo">

<xs:simpleType>

<xs:restriction base="xs:string">

<xs:length value="20"/>

</xs:restriction>

</xs:simpleType>

</xs:element>

<xs:element type="xs:string" name="wareHouseDeclarationItemNo">

<xs:simpleType>

<xs:restriction base="xs:string">

<xs:length value="3"/>

</xs:restriction>

</xs:simpleType>

</xs:element>

<xs:element type="xs:string" name="consignmentItemNo">

<xs:simpleType>

<xs:restriction base="xs:string">

<xs:length value="3"/>

</xs:restriction>

</xs:simpleType>

</xs:element>

<xs:element type="xs:decimal" name="amount">

<xs:simpleType>

<xs:restriction base="xs:decimal">

<xs:totalDigits value="4"/>

<xs:fractionDigits value="2"/>

</xs:restriction>

</xs:simpleType>

</xs:element>

<xs:element type="xs:string" name="amountUnit"/>

<xs:element type="xs:string" name="explanation">

<xs:simpleType>

<xs:restriction base="xs:string">

<xs:length value="100"/>

</xs:restriction>

</xs:simpleType>

</xs:element>

<xs:element type="xs:string" name="goodsAmount"/>

<xs:element type="xs:string" name="currencyType"/>

<xs:element type="xs:string" name="countryOfTrade"/>

<xs:element type="xs:string" name="deliveryType"/>

<xs:element type="xs:string" name="paymentType"/>

<xs:element type="xs:string" name="operationQualityCode"/>

</xs:sequence>

</xs:complexType>

</xs:element>

<xs:element name="goodsDispatchLocation" type="xs:string" nillable="true">

<xs:simpleType>

<xs:restriction base="xs:string">

<xs:length value="255"/>

</xs:restriction>

</xs:simpleType>

</xs:element>

<xs:element name="isDispatchUnderGIK117" type="xs:string">

<xs:simpleType>

<xs:restriction base="xs:string">

<xs:pattern value="[YN]"/>

</xs:restriction>

</xs:simpleType>

</xs:element>

<xs:element name="isSimplifiedProcedure" type="xs:string">

<xs:simpleType>

<xs:restriction base="xs:string">

<xs:pattern value="[YN]"/>

</xs:restriction>

</xs:simpleType>

</xs:element>

</xs:schema>

Gönderilen CCO15C mesajı veritabanına kaydedildikten sonra mesaj validasyon ve diğer kontroller otomatik olarak başlatılacaktır. Validasyon ve kontrol sonuçları ilgili GUID numarası ile veritabanına CC015\_RES mesajı olarak kaydedilecektir. CC015\_RES başarılı ve başarısız mesaj örnekleri aşağıdadır:

**Başarılı CC015\_RES Mesaj Örneği:**

<?xml version=\"1.0\" encoding=\"UTF-8\" standalone=\"yes\"?>

<ns2:CC015B\_RES xmlns:ns2=\"http://ncts.dgtaxud.ec\">

<GUID>24332423432423423</GUID>

<LRN>23LR000100000022</LRN>

<TMSTMP>Mon Sep 11 11:19:15 EET 2023</TMSTMP>

</ns2:CC015B\_RES>

**Başarısız CC015\_RES Mesaj Örneği:**

<?xml version=\"1.0\" encoding=\"UTF-8\" standalone=\"yes\"?>

<ns2:CC015B\_RES xmlns:ns2=\"http://ncts.dgtaxud.ec\">

<GUID>24332423432423423</GUID>

<TMSTMP>Mon Sep 11 11:26:42 EET 2023</TMSTMP>

<ERROR>[cvc-complex-type.2.4.a|CUSTOMSOFFICEOFDESTINATIONDECLARED|AUTHORISATION]</ERROR>

</ns2:CC015B\_RES>

### uploadMessageByLrn(String jsonQuery)

***Servis Path : “/trader/uploadMessageByLrn”***

Bu servis, daha önce gönderilmiş olan bir mesajı, CC013C mesajı göndererek değiştirmek için kullanılır.

**Servis İmzası:**

@POST  
@Path("/uploadMessageByLrn")  
@Consumes({MediaType.*APPLICATION\_JSON*})  
@Produces({MediaType.*APPLICATION\_JSON*})  
public Response uploadMessageByLrn(String jsonQuery)

**Servis Parametreleri:**

**“msgContent”** e-imzalı olarak gönderiliyor ise “signFlag” true olarak gönderilmesi ve “msgContent”’in imzalanmış veriyi içermesi gerekmektedir. İmzalanacak olan “msgContent”’in içeriği JSON objesi olarak aşağıdaki şekilde olmalıdır:

{

“phase5XML” : “xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx” (String),

“nationXML”: “xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx” (String)

}

**İmzalanmış olarak gönderilen mesaj içeriği:**

{

“token”: “xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx” (String),

“firmId”: “xxxxxxxxxxxxxxxxx” (String),

“userId”: “xxxxxxxxxxxxxx” (String),

“msgType”: ”CC013C” (String),

“signFlag”: true (Boolean),

“msgContent”: “xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx” (String, imzalanmış içerik)

}

**İmzalanmamış olarak gönderilen mesaj içeriği:**

{

“token”: “xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx” (String),

“firmId”: “xxxxxxxxxxxxxxxxx” (String),

“userId”: “xxxxxxxxxxxxxx” (String),

“msgType”: ”CC013C” (String),

“signFlag”: false (Boolean),

“msgContent”: {

“phase5XML” : “xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx” (String),

“nationXML”: “xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx” (String)

}

}

**Not: İmzalanmamış gönderim sadece test amaçlı olarak kullanılacaktır. Sistem canlıya alındığında imzalanmamış içeriğin gönderimine izin verilmeyecektir**

**Servis Dönüş Parametreleri (Başarılı) :**

{

"data" : {

"guid" : "859e99e2-25f9-4b64-ace2-2f5bb1f4d65f"

},

"error" : [ ],

"status" : 200

}

**Servis Dönüş Parametreleri (Başarısız) :**

{

"data" : null,

"error" : [ "Parameters are not valid. Please check" ],

"status" : 400

}

**Örnek Kullanımı:**

public void testUploadMessageByLrn () throws Exception {  
 WSCommunicator wsCommunicator = new WSCommunicator("http://bilge.ncts.traderportal/trader/");  
 byte[] xmlBytes = Files.*readAllBytes*(Paths.*get*("C:\\NCTS\_XML\_FILES\\message.xml"));  
 String xmlMessage = new String(xmlBytes, StandardCharsets.*UTF\_8*);  
 xmlBytes = Files.*readAllBytes*(Paths.*get*("C:\\NCTS\_XML\_FILES\\nation.xml"));  
 String nationMessage = new String(xmlBytes, StandardCharsets.*UTF\_8*);  
 JsonObject payload = new JsonObject();  
 JsonObject msgContent = new JsonObject();  
 msgContent.put(GeneralConstants.*TRADER\_SRV\_PHASE5\_XML*, xmlMessage);  
 msgContent.put(GeneralConstants.*TRADER\_SRV\_NATION\_XML*, nationMessage);  
  
 payload.put(GeneralConstants.*TRADER\_SRV\_TOKEN*, getToken());  
 payload.put(GeneralConstants.*TRADER\_SRV\_FIRM\_ID*, "MilSOFT");  
 payload.put(GeneralConstants.*TRADER\_SRV\_USER\_ID*, "eserin");

payload.put(GeneralConstants.*TRADER\_SRV\_LRN*, " 23LR061600000008");  
 payload.put(GeneralConstants.*TRADER\_SRV\_MSG\_TYPE*, "CC013C");  
 payload.put(GeneralConstants.*TRADER\_SRV\_SIGN\_FLAG*, false);  
 payload.put(GeneralConstants.*TRADER\_SRV\_MSG\_CONTENT*, msgContent);  
 String result = wsCommunicator.postData(payload, "uploadMessageByLrn");  
 JsonObject resultObject = new JsonObject(result);  
 System.*out*.println(resultObject.encodePrettily());  
}

### getMessageListByGuid(String jsonQuery)

***Servis Path : “/trader/getMessageListByGuid”***

Bu servis, belirtilen “guid” ile ilgili oluşan tüm mesajların listesinin dönülmesi için kullanılır.

**Servis İmzası:**

@POST  
@Path("/getMessagesListByGuid")  
@Consumes({MediaType.*APPLICATION\_JSON*})  
@Produces({MediaType.*APPLICATION\_JSON*})  
public Response getMessagesListByGuid(String jsonQuery)

**Servis Parametreleri:**

{

“token”: “xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx” (String),

“firmId”: “xxxxxxxxxxxxxxxxx” (String) ,

“userId”: “xxxxxxxxxxxxxx” (String),

“guid”: “xxxxxxxxxxxxxxxxxx” (String),

}

**Servis Dönüş Parametreleri:**

{

"data" : {

messages: [

{“msgIndex” : xxxxxx (Long), “msgType”: “xxxxxxx” (String), “msgStatus” : “String” (READ|UNREAD), “guid”:”xxxxxxxxx” (String), “lrn”: “xxxxxxx” (String), “mrn”: “xxxxx” (String)},

{“msgIndex” : xxxxxx (Long), “msgType”: “xxxxxxx” (String), “msgStatus” : “String” (READ|UNREAD) ), “guid”:”xxxxxxxxx” (String), “lrn”: “xxxxxxx” (String), “mrn”: “xxxxx” (String)},

{“msgIndex” : xxxxxx (Long), “msgType”: “xxxxxxx” (String), “msgStatus” : “String” (READ|UNREAD) ), “guid”:”xxxxxxxxx” (String), “lrn”: “xxxxxxx” (String), “mrn”: “xxxxx” (String)}

]

},

"error" : [ ],

"status" : 200

}

**Örnek Kullanımı:**

public void getMessageListByGuid () throws Exception {  
 WSCommunicator wsCommunicator = new WSCommunicator("http://bilge.ncts.traderportal/trader/");  
 JsonObject payload = new JsonObject();

payload.put(GeneralConstants.*TRADER\_SRV\_TOKEN*, getToken());  
 payload.put(GeneralConstants.*TRADER\_SRV\_FIRM\_ID*, "MilSOFT");  
 payload.put(GeneralConstants.*TRADER\_SRV\_USER\_ID*, "eserin");

payload.put(GeneralConstants.*GUIİD*, " e3f7cb90-47bf-4034-99f3-b53efc26b41c");  
 String result = wsCommunicator.postData(payload, "getMessageListByGuid");  
 JsonObject resultObject = new JsonObject(result);  
 System.*out*.println(resultObject.encodePrettily());  
}

### getUnReadMessageList(String jsonQuery)

***Servis Path : “/trader/getUnReadMessageList”***

Bu servis, okunmayan tüm mesajların listesinin dönülmesi için kullanılır.

**Servis İmzası:**

@POST  
@Path("/getUnReadMessagesList")  
@Consumes({MediaType.*APPLICATION\_JSON*})  
@Produces({MediaType.*APPLICATION\_JSON*})  
public Response getUnReadMessagesList(String jsonQuery) {

**Servis Parametreleri:**

{

“token”: “xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx” (String),

“firmId”: “xxxxxxxxxxxxxxxxx” (String) ,

“userId”: “xxxxxxxxxxxxxx” (String),

}

**Servis Dönüş Parametreleri:**

{

"data" : {

messages: [

{“msgIndex” : xxxxxx (Long), “msgType”: “xxxxxxx” (String) ), “guid”:”xxxxxxxxx” (String), “lrn”: “xxxxxxx” (String), “mrn”: “xxxxx” (String) },

{“msgIndex” : xxxxxx (Long), “msgType”: “xxxxxxx” (String) ), “guid”:”xxxxxxxxx” (String), “lrn”: “xxxxxxx” (String), “mrn”: “xxxxx” (String) },

{“msgIndex” : xxxxxx (Long), “msgType”: “xxxxxxx” (String) ), “guid”:”xxxxxxxxx” (String), “lrn”: “xxxxxxx” (String), “mrn”: “xxxxx” (String)}

]

},

"error" : [ ],

"status" : 200

}

**Örnek Kullanımı:**

public void getUnReadMessageList () throws Exception {  
 WSCommunicator wsCommunicator = new WSCommunicator("http://bilge.ncts.traderportal/trader/");  
 JsonObject payload = new JsonObject();

payload.put(GeneralConstants.*TRADER\_SRV\_TOKEN*, getToken());  
 payload.put(GeneralConstants.*TRADER\_SRV\_FIRM\_ID*, "MilSOFT");  
 payload.put(GeneralConstants.*TRADER\_SRV\_USER\_ID*, "eserin");

String result = wsCommunicator.postData(payload, "getUnReadMessageList");  
 JsonObject resultObject = new JsonObject(result);  
 System.*out*.println(resultObject.encodePrettily());  
}

### downloadMessageByIndex(String jsonQuery)

***Servis Path : “/trader/downloadMessageByIndex”***

Bu servis, index’i verilen mesajın dönülmesini sağlar.

**Servis İmzası:**

@POST  
@Path("/downloadMessageByIndex")  
@Consumes({MediaType.*APPLICATION\_JSON*})  
@Produces({MediaType.*APPLICATION\_JSON*})  
public Response downloadMessageByIndex(String jsonQuery)

**Servis Parametreleri:**

{

“token”: “xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx” (String),

“firmId”: “xxxxxxxxxxxxxxxxx” (String) ,

“userId”: “xxxxxxxxxxxxxx” (String),

“msgIndex”: xxxxxxxxxx (Long),

}

**Servis Dönüş Parametreleri :**

{

"data" : {

“msgType”: ”xxxxxx” (String),

“xmlContent”: “xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx” (String),

“guid”:”xxxxxxxxx” (String), “lrn”: “xxxxxxx” (String), “mrn”: “xxxxx” (String)

},

"error" : [ ],

"status" : 200

}

**Örnek Kullanımı:**

public void downloadMessageByIndex () throws Exception {  
 WSCommunicator wsCommunicator = new WSCommunicator("http://bilge.ncts.traderportal/trader/");  
 JsonObject payload = new JsonObject();

payload.put(GeneralConstants.*TRADER\_SRV\_TOKEN*, getToken());  
 payload.put(GeneralConstants.*TRADER\_SRV\_FIRM\_ID*, "MilSOFT");  
 payload.put(GeneralConstants.*TRADER\_SRV\_USER\_ID*, "eserin");

payload.put(GeneralConstants.*TRADER\_SRV\_INDEX*, 12345);

String result = wsCommunicator.postData(payload, "downloadMessageByIndex");  
 JsonObject resultObject = new JsonObject(result);  
 System.*out*.println(resultObject.encodePrettily());  
}

### getLrnByGuid(String jsonQuery)

***Servis Path : “/trader/getLrnByGuid”***

Bu servis, belirtilen guid için oluşturulan LRN numarasının dönülmesini sağlar.

**Servis İmzası:**

@POST  
@Path("/getLrnByGuid")  
@Consumes({MediaType.*APPLICATION\_JSON*})  
@Produces({MediaType.*APPLICATION\_JSON*})  
public Response getLrnByGuid(String jsonQuery)

**Servis Parametreleri:**

{

“token”: “xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx” (String),

“firmId”: “xxxxxxxxxxxxxxxxx” (String) ,

“userId”: “xxxxxxxxxxxxxx” (String),

“guid”: “e3f7cb90-47bf-4034-99f3-b53efc26b41c” (String)

}

**Servis Dönüş Parametreleri :**

{

"data" : {

“lrn”: ”xxxxxxxxxxxxxx” (String)

},

"error" : [ ],

"status" : 200

}

**Örnek Kullanımı:**

public void getLrnByGuid () throws Exception {  
 WSCommunicator wsCommunicator = new WSCommunicator("http://bilge.ncts.traderportal/trader/");  
 JsonObject payload = new JsonObject();

payload.put(GeneralConstants.*TRADER\_SRV\_TOKEN*, getToken());  
 payload.put(GeneralConstants.*TRADER\_SRV\_FIRM\_ID*, "MilSOFT");  
 payload.put(GeneralConstants.*TRADER\_SRV\_USER\_ID*, "eserin");

payload.put(GeneralConstants.*TRADER\_SRV\_GUID*, “e3f7cb90-47bf-4034-99f3-b53efc26b41c”);

String result = wsCommunicator.postData(payload, "getLrnByGuid");  
 JsonObject resultObject = new JsonObject(result);  
 System.*out*.println(resultObject.encodePrettily());  
}

### getMrnByGuid(String jsonQuery)

***Servis Path : “/trader/getMrnByGuid”***

Bu servis, belirtilen guid için oluşturulan MRN numarasının dönülmesini sağlar.

**Servis İmzası:**

@POST  
@Path("/getMrnByGuid")  
@Consumes({MediaType.*APPLICATION\_JSON*})  
@Produces({MediaType.*APPLICATION\_JSON*})  
public Response getMrnByGuid(String jsonQuery)

**Servis Parametreleri:**

{

“token”: “xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx” (String),

“firmId”: “xxxxxxxxxxxxxxxxx” (String) ,

“userId”: “xxxxxxxxxxxxxx” (String),

“guid”: “e3f7cb90-47bf-4034-99f3-b53efc26b41c” (String)

}

**Servis Dönüş Parametreleri :**

{

"data" : {

“mrn”: ”xxxxxxxxxxxxxx” (String)

},

"error" : [ ],

"status" : 200

}

**Örnek Kullanımı:**

public void getMrnByGuid () throws Exception {  
 WSCommunicator wsCommunicator = new WSCommunicator("http://bilge.ncts.traderportal/trader/");  
 JsonObject payload = new JsonObject();

payload.put(GeneralConstants.*TRADER\_SRV\_TOKEN*, getToken());  
 payload.put(GeneralConstants.*TRADER\_SRV\_FIRM\_ID*, "MilSOFT");  
 payload.put(GeneralConstants.*TRADER\_SRV\_USER\_ID*, "eserin");

payload.put(GeneralConstants.*TRADER\_SRV\_GUID*, “e3f7cb90-47bf-4034-99f3-b53efc26b41c”);

String result = wsCommunicator.postData(payload, "getMrnByGuid");  
 JsonObject resultObject = new JsonObject(result);  
 System.*out*.println(resultObject.encodePrettily());  
}

### getVersion ()

***Servis Path : “/trader/getVersion”***

Bu servis, yazılımın versiyonunu almak için kullanılır.

**Servis İmzası:**

@GET  
@Path("/getVersion")  
@Consumes({MediaType.*APPLICATION\_JSON*})  
@Produces({MediaType.*APPLICATION\_JSON*})  
public Response getVersion()

**Servis Dönüş Parametreleri:**

{

"data" : {

“version”: ”xxxxxxxxxxxxxx” (String)

},

"error" : [ ],

"status" : 200

}

**Örnek Kullanımı:**

public void getVersion () throws Exception {  
 WSCommunicator wsCommunicator = new WSCommunicator("http://bilge.ncts.traderportal/trader/");  
 JsonObject payload = new JsonObject();

String result = wsCommunicator.getData(payload, "getVersion");  
 JsonObject resultObject = new JsonObject(result);  
 System.*out*.println(resultObject.encodePrettily());  
}