**SK m&service Benepia**

**베네피아 인증 서버 ­­연동 규격서**

*(SSO 인증 토큰 API 연동 - 제휴사)*

**Version 1.0.1**

2019-09-03



서울특별시 중구 퇴계로 385 준타워



사용 권한

본 문서에 대한 서명은 SK m&service 내부에서 본 문서에 대하여 수행 및 유지관리의 책임이 있음을 인정하는 것입니다.

|  |  |  |  |
| --- | --- | --- | --- |
| 작성자 : | SK m&service 장수환 매니저 | 일자 : | 2017-06-22 |
| 승인자 : | SK m&service 박진완 팀장 | 일자 : | 2017-06-22 |

Revisions

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 버전 | 변경일자 | Revisions | 작성자 | 이메일 |
| 1.0.1 | 2019-09-03 | * 개발서버 URL 변경 * 사전 작업 (방화벽, 접근 IP) 추가 | 홍수명 | s.m.hong@sk.com |
| 1.0.0 | 2017-06-22 | * 베네피아 SSO 인증 토큰 API 연동 규격서 초안 | 장수환 | [jshwan777@sk.com](mailto:jshwan777@sk.com) |

Spec. Release

본 문서는 베네피아와 제휴사간의 SSO(Single Sign On) 처리를 위해 SSO 인증 토큰 API 사용에 대한 전반적인 설명과 연동 규약에 대해 기술합니다.

References

|  |  |  |
| --- | --- | --- |
| **[RFC2119]** | “Key words for use in RFCs to Indicate Requirement Levels”. S. Bradner. March 1997. <URL:http://www.ietf.org/rfc/rfc2119.txt> | |
| **[HTTP/1.1]** | | Hyper Text Transfer Protocol (HTTP/1.1), <http://www.ietf.org/rfc/rfc2616.txt> |
| **[HTTM SM]** | | The HTTP State Management Mechanism, <http://www.ietf.org/rfc/rfc6265.txt> |
| **[URI]** | | Uniform Resource Identifier (URI): Generic Syntax, <http://www.ietf.org/rfc/rfc2396.txt> |
| **[SEED]** | | The SEED Encryption Algorithm, **<https://tools.ietf.org/html/rfc4269>** |
| **[BASE64]** | | The Base16, Base32, and Base64 Data Encodings, [**https://tools.ietf.org/html/rfc4648**](https://tools.ietf.org/html/rfc4648) |
| **[JSON]** | | The application/json Media Type for JavaScript Object Notation (JSON), [**http://www.ietf.org/rfc/rfc4627.txt**](http://www.ietf.org/rfc/rfc4627.txt) |
| **[XML]** | | The application/xml Media Type [**http://www.ietf.org/rfc/rfc3023.txt**](http://www.ietf.org/rfc/rfc3023.txt) |

Introduction

본 문서는 베네피아 SSO 인증 서버를 통해 베네피아 복지몰에서 제휴사로 이동시 SSO(Single Sign On) 연동을 하기 위해 Server to Server간의 API연동 규격을 정의하고 요청 및 응답에 대한 Sample Codes를 포함합니다.

암호화에 필요한 SEED secret key와 iv(initial vector)는 별도의 이메일을 통해서 전달 합니다.

# 용어정리

## 제휴사

베네피아 복지몰 서비스내에 이용하고 있는 회사를 의미합니다.

## 제휴사Server

제휴사에서 사용하는 Server가 이에 해당됩니다.

## 제휴사 코드

제휴사마다 별도로 부여된 4자리의 유니크한 코드 값입니다.

: (이메일을 통해 제휴사별 코드값 전달)

## BASE64

암호화 후 바이너리를 문자열로 처리하기 위한 방법입니다.

# 사전 작업

## 방화벽

베네피아 운영은 ANY로 OPEN 되어 있어 제휴사 서버에서 Outbound 방화벽만 작업하면 됩니다.

개발기는 별도의 양사간 방화벽 작업을 진행해야 합니다. (베네피아\_SSO\_개발\_방화벽\_(제휴사명).xlsx)

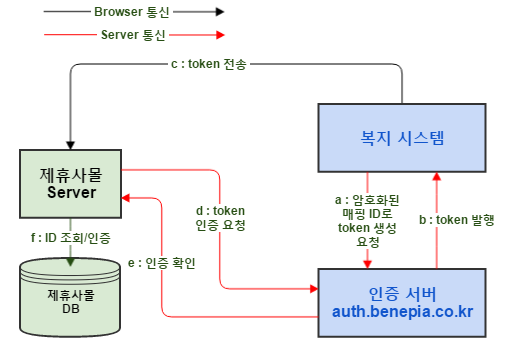
## IP 접근 차단

인증서버는 별도의 방화벽 작업(개발제외)을 하지 않지만 접근하는 IP에 대한 관리는 하고 있습니다.

Server To Server 통신으로 처리하기에 개발/운영의 Real IP / Nat IP를 전달해 주셔야 합니다.

# Workflow

베네피아 서비스의 SSO 인증을 위한 토큰 발행 및 확인하는 흐름도를 간단하게 설명합니다.



**☞복지 시스템으로부터 전송받은 token(c:토큰전송) 데이터를 베네피아 인증서버를 통해 유효한 토큰인지 인증 확인 요청(d:token 인증 요청) 후 인증 확인에 성공하면 제휴사 몰로 진입 (Server to Server 통신)**

* HTTP POST방식으로 연동합니다.
* 베네피아 인증 서버 정보

|  |  |  |  |
| --- | --- | --- | --- |
| 구분 | IP | URL | PORT |
| 운영 | 118.219.59.150 | <https://auth.benepia.co.kr> | 443 |
| 개발 | 118.219.59.205 | <https://stg-auth.benepia.co.kr> | 443 |

* API 목록

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Operation | API | Response Format |
| 1 | Token Key 인증 확인 요청 | <https://auth.benepia.co.kr/api/token/confirm> | JSON, XML |
|  |  |  |  |

# Getting Started

베네피아 토큰 인증 API 연동으로 제휴사에서는 아래 내용을 참고하여 연동 개발을 진행하면 됩니다.

# API 인터페이스 정의

## Token Key 확인 요청(Confirm)

### 요청방식

* + HTTP  request
* Header - Request Method : **POST**
* Content-Type : **application/x-www-form-urlencoded**

### Request URL

* + <https://auth.benepia.co.kr/api/token/confirm.json>
* <https://auth.benepia.co.kr/api/token/confirm.xml>

### Request Parameter

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field Name | Type | description | value | Size |
| tknKey | String | 전송 받은 Token Key | d95e223f768adb932c7a38b7dc403de9bf5cfb6ea22108ad6cd0fe7b42d45840087442f6 | 72 |
| custCoCd | String | 제휴사코드 | 00XX  (베네피아 담당자 발급) | 4 |

### Response Parameter

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Field Name | type | description | value | Size |
| responseCode | String | 응답 코드(별도 응답 코드 표 참조) | S000 : 성공, E00X:에러… | 4 |
| responseMessage | String | 응답 메시지 |  | 46 |
| additionalMessage | String | 추가 메시지 |  |  |
| responseData | Child | 전송받은 Token Key | tknKey | 72 |
| 클라이언트 IP | clientIp | 15 |

### Response JSON Format

|  |
| --- |
| Header      'content-type' - application/json      'status' - 200  --------------------------------------------------------------------------------------------------  Body  - 성공시  {      "responseCode": "S000",      "responseMessage": "성공",      "additionalMessage": "토큰 정보가 확인되었습니다.",      "responseData":{          "tknKey": "d95e223f768adb932c7a38b7dc403de9bf5cfb6ea22108ad6cd0fe7b42d45840087442f6",          "clientIp": "127.0.0.1"      }  }  - 실패  {      "responseCode": "E001",      "responseMessage": "올바르지 않은 요청입니다.",      "additionalMessage": "",      "responseData":{}  }  - 실패  {      "responseCode": "E002",      "responseMessage": "올바르지 않은 파라미터입니다.",      "additionalMessage": "",      "responseData":{}  }  - 실패  {      "responseCode": "E003",      "responseMessage": "토큰이 만료되었습니다.",      "additionalMessage": "",      "responseData":{}  }  - 실패  {      "responseCode": "E004",      "responseMessage": "토큰 조회 허용 횟수를 초과하였습니다.",      "additionalMessage": "",      "responseData":{}  }  - 실패  {      "responseCode": "E005",      "responseMessage": "허용되지 않은 IP입니다.",      "additionalMessage": "",      "responseData":{}  } |

### Response XML Format

|  |
| --- |
| Header      'content-type' - application/xml      'status' - 200  ---------------------------------------------------------------------------------------------------  Body  - 성공  <?xml version="1.0" encoding="UTF-8" standalone="yes" ?>  <response>      <responseCde>S000</responseCode>      <responseMessage>성공</responseMessage>      <additionalMessage>토큰 정보가 확인되었습니다.</additionalMessage>      <responseData>          <clientIp>127.0.0.1</clientIp>          <tknKey>394ffe57f67dc98d1d27857c40369e09b797ffecef21f6b577d4010eaaab44398386fd13</tknKey>      </responseData>  </response>    - 실패  <?xml version="1.0" encoding="UTF-8" standalone="yes" ?>  <response>      <responseCde>E001</responseCode>      <responseMessage>올바르지 않은 요청입니다.</responseMessage>      <additionalMessage/>  </response>    - 실패  <?xml version="1.0" encoding="UTF-8" standalone="yes" ?>  <response>      <responseCde>E002</responseCode>      <responseMessage>올바르지 않은 파라미터입니다.</responseMessage>      <additionalMessage/>  </response>  - 실패  <?xml version="1.0" encoding="UTF-8" standalone="yes" ?>  <response>      <responseCde>E003</responseCode>      <responseMessage>토큰이 만료되었습니다.</responseMessage>      <additionalMessage/>  </response>  - 실패  <?xml version="1.0" encoding="UTF-8" standalone="yes" ?>  <response>      <responseCde>E004</responseCode>      <responseMessage>토큰 조회 허용 횟수를 초과 하였습니다.</responseMessage>      <additionalMessage/>  </response>  - 실패  <?xml version="1.0" encoding="UTF-8" standalone="yes" ?>  <response>      <responseCde>E005</responseCode>      <responseMessage>허용되지 않은 IP입니다.</responseMessage>      <additionalMessage/>  </response> |

# 응답 코드 정리

|  |  |
| --- | --- |
| ResponseCode | Description |
| S000 | 성공 |
| E000 | 시스템에러 |
| E001 | 올바르지 않은 요청입니다. |
| E002 | 올바르지 않은 파라미터입니다. |
| E003 | 토큰이 만료되었습니다. |
| E004 | 토큰 조회 허용 횟수를 초과 하였습니다. |
| E005 | 허용되지 않은 IP입니다. |
| E006 | 인증에 실패했습니다. 담당자에게 문의 바랍니다. |
| -1000 | 서버 점검 중입니다. |

# #. Appendix

* **언어별 Sample Code**

연동규격과 더불어 언어별(Java, PHP, C#, ASP) Sample Code를 참고해주세요

1. **Java Sample 코드**

※ custCoCd의 5555는 예제입니다. 제휴사에 맞는 코드로 등록하셔야 합니다.

|  |
| --- |
| **import** java.io.BufferedReader;  **import** java.io.DataOutputStream;  **import** java.io.InputStream;  **import** java.io.InputStreamReader;  **import** java.net.HttpURLConnection;  **import** java.net.URL;  **import** java.util.HashMap;  **import** java.util.Map;  **import** javax.net.ssl.HttpsURLConnection;  **import** javax.xml.parsers.DocumentBuilder;  **import** javax.xml.parsers.DocumentBuilderFactory;  **import** org.w3c.dom.Document;  StringBuffer sb = **new** StringBuffer();  // 반드시 파라미터는 반드시 인코딩 필요  sb.append("tknKey="+URLEncoder.encode("89936bb2a16158f320af1f5279437c97b2a05e0ef0dd5898a14248a15d8727913ac88568", "UTF-8") + "&");  sb.append("custCoCd="+URLEncoder.encode("5555", "UTF-8"));  URL url = **new** URL("https://auth.benepia.co.kr/api/token/confirm.json"); //JSON포멧으로 요청  //URL url = **new** URL("https://auth.benepia.co.kr/api/token/ confirm.xml"); //XML포멧으로 요청  HttpsURLConnection conn = (HttpsURLConnection)url.openConnection();  conn.setRequestMethod("POST");  conn.setDoInput(**true**);  conn.setDoOutput(**true**);  conn.setUseCaches(**false**);  conn.setDefaultUseCaches(**false**);  DataOutputStream wr = **new** DataOutputStream(conn.getOutputStream());  wr.writeBytes(sb.toString());  wr.flush();  wr.close();  StringBuffer buffer = **new** StringBuffer();  **int** status = conn.getResponseCode();  //200이 아니면 전부 에러 처리  **if**(status != 200) {  // 에러 처리  }  BufferedReader getData = **new** BufferedReader( **new** InputStreamReader(conn.getInputStream(),"UTF-8"));  String line;  **while**((line = getData.readLine()) != **null**) {  buffer.append(line);  }  getData.close();  buffer.toString();  ----------------------------------------------------------------------------------------------------  //JSON으로 요청했을 경우 JSON Format 처리  String jsonString = buffer.toString();  //ObjectMapper를 이용해 JSON데이터를 Java Object로 변환  ObjectMapper mapper = **new** ObjectMapper();  RootResponseVO rootResponseVO = mapper.readValue(jsonString, RootResponseVO.**class**);  //XML로 요청했을 경우 XML Format 처리  String xmlString = buffer.toString();  //Unmarshaller를 이용해 XML데이터를 Java Object로 변환  JAXBContext jaxbContext = JAXBContext.newInstance(CommonResponseVO.**class**);  Unmarshaller jaxbUnmarshaller = jaxbContext.createUnmarshaller();  InputSource is = **new** InputSource();  is.setCharacterStream(**new** StringReader(xmlString));  RootResponseVO rootResponseVo = (RootResponseVO) jaxbUnmarshaller.unmarshal(is); |

1. **PHP Sample 코드**

|  |
| --- |
| //xml  $url = 'https://auth.benepia.co.kr/api/token/confirm.xml';  //json  //$url = 'https://authdev.benepia.co.kr/api/token/confirm.json';    // POST로 전송할 Parameter 설정  $postParam = array(  'tknKey'=>urlencode('89936bb2a16158f320af1f5279437c97b2a05e0ef0dd5898a14248a15d8727913ac88568'),  'custCoCd'=>urlencode('5555')  );    $ch = curl\_init();  curl\_setopt($ch, CURLOPT\_URL, $url);  curl\_setopt($ch, CURLOPT\_SSL\_VERIFYPEER, FALSE);  curl\_setopt($ch, CURLOPT\_SSL\_VERIFYHOST, 0);  curl\_setopt($ch, CURLOPT\_HEADER, 0);  curl\_setopt($ch, CURLOPT\_USERAGENT,$\_SERVER['HTTP\_USER\_AGENT']);  curl\_setopt($ch, CURLOPT\_RETURNTRANSFER, 1);  curl\_setopt($ch, CURLOPT\_POST, 1);  curl\_setopt($ch, CURLOPT\_POSTFIELDS, $postParam);    $data = curl\_exec($ch);    curl\_close($ch);  $decodeStr = json\_decode($data, **true**);  print\_r($decodeStr); |

1. **C# Sample 코드**

|  |
| --- |
| string url = "https://auth.benepia.co.kr/api/token/confirm.xml";  // POST로 전송할 Parameter 설정  StringBuilder postParam = **new** StringBuilder();  postParam.Append("tknKey=");  postParam.Append(Server.UrlEncode("89936bb2a16158f320af1f5279437c97b2a05e0ef0dd5898a14248a15d8727913ac88568"));  postParam.Append("&custCoCd=");  postParam.Append(Server.UrlEncode("5555"));  // 요청 String -> 요청 Byte 변환  **byte**[] byteDataParams = UTF8Encoding.UTF8.GetBytes(postParam.ToString());    // HttpWebRequest 객체 생성, 설정  HttpWebRequest request = (HttpWebRequest)WebRequest.Create(url);  request.Method = "POST";  request.ContentType = "application/x-www-form-urlencoded";  request.ContentLength = byteDataParams.Length;    // 요청 Byte -> 요청 Stream 변환  Stream stDataParams = request.GetRequestStream();  stDataParams.Write(byteDataParams, 0, byteDataParams.Length);  stDataParams.Close();    // 요청, 응답 받기  HttpWebResponse response = (HttpWebResponse)request.GetResponse();    // 응답 Stream 읽기  Stream stReadData = response.GetResponseStream();  StreamReader srReadData = **new** StreamReader(stReadData, Encoding.Default);    // 응답 Stream -> 응답 String 변환  string strResult = srReadData.ReadToEnd();    Console.WriteLine(strResult);  Console.ReadLine(); |

1. **ASP Sample 코드**

|  |
| --- |
| Dim oXMLHTTP  Dim strStatusTest  Dim postParam    postParam = "tknKey="  postParam=postParam&Server.URLEncode("89936bb2a16158f320af1f5279437c97b2a05e0ef0dd5898a14248a15d8727913ac88568");  postParam = postParam & "&custCoCd="  postParam = postParam & Server.URLEncode("5555");  Set oXMLHTTP = CreateObject("MSXML2.ServerXMLHTTP.3.0")    oXMLHTTP.Open "POST", "https://auth.benepia.co.kr/api/token/confirm.xml", False  oXMLHTTP.setRequestHeader "Content-Type", "application/x-www-form-urlencoded"  oXMLHTTP.Send postParam    If oXMLHTTP.Status = 200 Then  GetTextFromUrl = oXMLHTTP.responseText  End If |

- End of Documentation -