



Authentication and Data Samples for New Hire Interconnection

Integration security and the protection of information are of the highest priority. To support organization integration, USA Staffing incorporates the following:

- **Secure Connection.** All integration communication will be HTTPS over TLS. The USA Staffing Program Office and the HR Tools and Technology team will work with each integration partner to ensure the appropriate version is utilized.
- **IP Address Validation.** For each JWT request generated from USAS, the IP address is validated against the Authenticate Token to verify the message as authentically being from an approved Organization Partner. Each Organization authorized (Partner) will have a defined TCP/IP range of addresses which are considered valid. For communication to occur through the New Hire Interconnection, the combination of IP address and JWT must be valid.

JSON Web Token (JWT)

JSON Web Token (JWT) is an open standard ([RFC 7519](#)) that defines a compact and self-contained way for securely transmitting information between parties as a JSON object. This information can be verified and trusted because it is digitally signed. JWTs can be signed using a secret (with the HMAC algorithm) or a public/private key pair using RSA.

Because of their smaller size, JWTs can be sent through a URL, POST parameter, or inside an HTTP header. Additionally, the smaller size means transmission is fast. The payload contains all the required information about the user, avoiding the need to query the database more than once.

The benefits of JWT when compared to Simple Web Tokens (SWT) and Security Assertion Markup Language Tokens (SAML).

As JSON is less verbose than XML, when it is encoded its size is also smaller, making JWT more compact than SAML. This makes JWT a good choice to be passed in HTML and HTTP environments.

Security-wise, SWT can only be symmetrically signed by a shared secret using the HMAC algorithm. However, JWT and SAML tokens can use a public/private key pair in the form of a X.509 certificate for signing. Signing XML with XML Digital Signature without introducing obscure security holes is very difficult when compared to the simplicity of signing JSON.

Comparison of the length of an encoded JWT and an encoded SAML

The screenshot shows the 'JSON Web Tokens - jwt.io' browser window. The 'Encoded' section contains a long, single-line JWT string: `eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoiYWRtaW4iOnRydWV9.TjVA9SOrM7E2cBab30RMRhHDcEfxjoYZgeFONFh7HgQ`. The 'Decoded' section shows the header: `{ "alg": "HS256", "typ": "JWT" }`, the payload: `{ "sub": "1234567890", "name": "John Doe", "admin": true }`, and the signature verification details for HMACSHA256, including the formula: `base64UrlEncode(header) + "." + base64UrlEncode(payload), secret`.



The screenshot shows the 'SAML - samtool.io' browser window. The 'SAML ENCODED' section contains a very long, multi-line base64-encoded SAML response. The 'SAML DECODED' section shows the corresponding XML structure, including elements like `<saml:Response>`, `<saml:Issuer>`, `<saml:Assertion>`, `<Signature>`, and `<SignedInfo>`. The XML is displayed with line numbers from 1 to 26.

Authentication code sample

```
namespace AgencySample.Controllers
{
    public class AuthenticationController : ApiController
    {
        private const string Secret = "856FECBA3B06519C8DDDBC80BB080553";
        public HttpResponseMessage Post([FromBody]JObject value)
        {
            var response = Request.CreateResponse(HttpStatusCode.Forbidden);
            string content = "";
            if (value.GetValue("userName").Value<string>().ToUpper() == "testName".ToUpper() &&
                value.GetValue("password").Value<string>() == "P@ssword!")
            {
                response = Request.CreateResponse(HttpStatusCode.OK);
                JObject jwtToken = new JObject();
                jwtToken.Add("token", getJwtToken(value.GetValue("userName").Value<string>()));
                content = jwtToken.ToString();
            }
            response.Content = new StringContent(content, Encoding.UTF8, "application/json");
            return response;
        }

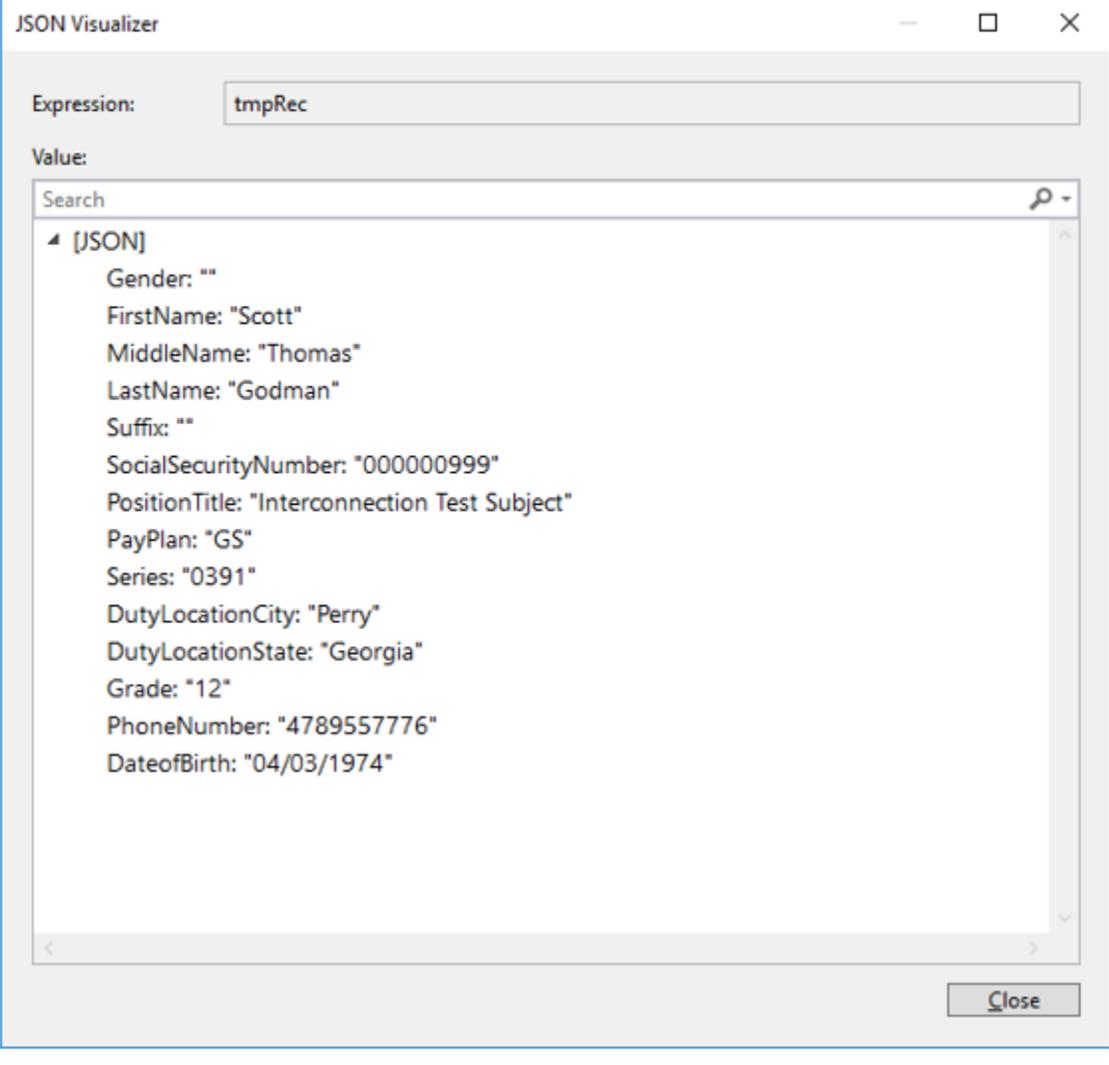
        private string getJwtToken(string userName)
        {
            var key = Convert.FromBase64String(Secret);
            var tokenHandler = new JwtSecurityTokenHandler();
            var tokenDescriptor = new SecurityTokenDescriptor
            {
                Subject = new ClaimsIdentity(new[]
                {
                    new Claim(ClaimTypes.Name, userName)
                }
                ),
                Expires = DateTime.Now.AddMinutes(Convert.ToInt32(600)),
                SigningCredentials = new SigningCredentials(new SymmetricSecurityKey(key), SecurityAlgorithms.HmacSha256Signature)
            };
            var desc = tokenHandler.CreateToken(tokenDescriptor);

            return tokenHandler.WriteToken(desc);
        }

        public static bool ValidateToken(string token)
        {
            return true;
        }
    }
}
```

Sample Data from USA Staffing Onboarding

```
{FirstName:"Scott","MiddleName":"Thomas","LastName":"Godman","Suffix":null,"SocialSecurityNumber":"000000999","PositionTitle":"Interconnection Test Subject","PayPlan":"GS","Series":"0391","DutyLocationCity":"Perry","DutyLocationState":"Georgia","Grade":"12","PhoneNumber":"4789557776","DateofBirth":"04/03/1974"}
```



The screenshot shows a window titled "JSON Visualizer" with a search bar and a "Close" button. The "Expression" field contains "tmpRec". The "Value" field displays the following JSON object:

```
{  
  "Gender": "",  
  "FirstName": "Scott",  
  "MiddleName": "Thomas",  
  "LastName": "Godman",  
  "Suffix": "",  
  "SocialSecurityNumber": "000000999",  
  "PositionTitle": "Interconnection Test Subject",  
  "PayPlan": "GS",  
  "Series": "0391",  
  "DutyLocationCity": "Perry",  
  "DutyLocationState": "Georgia",  
  "Grade": "12",  
  "PhoneNumber": "4789557776",  
  "DateofBirth": "04/03/1974"  
}
```

JSON Web Token Process

