



Verifying XML Signatures

Technical Note

UPDATED: 27 July 2023

© 2022 Progress Software Corporation and/or one of its subsidiaries or affiliates. All rights reserved.

These materials and all Progress® software products are copyrighted and all rights are reserved by Progress Software Corporation. The information in these materials is subject to change without notice, and Progress Software Corporation assumes no responsibility for any errors that may appear therein. The references in these materials to specific platforms supported are subject to change.

#1 Load Balancer in Price/Performance, 360 Central, 360 Vision, Chef, Chef (and design), Chef Habitat, Chef Infra, Code Can (and design), Compliance at Velocity, Corticon, Corticon.js, DataDirect (and design), DataDirect Cloud, DataDirect Connect, DataDirect Connect64, DataDirect XML Converters, DataDirect XQuery, DataRPM, Defrag This, Deliver More Than Expected, DevReach (and design), Driving Network Visibility, Flowmon, Inspec, Ipswitch, iMacros, K (stylized), Kemp, Kemp (and design), Kendo UI, Kinvey, LoadMaster, MessageWay, MOVEit, NativeChat, OpenEdge, Powered by Chef, Powered by Progress, Progress, Progress Software Developers Network, SequeLink, Sitefinity (and Design), Sitefinity, Sitefinity (and design), Sitefinity Insight, SpeedScript, Stylized Design (Arrow/3D Box logo), Stylized Design (C Chef logo), Stylized Design of Samurai, TeamPulse, Telerik, Telerik (and design), Test Studio, WebSpeed, WhatsConfigured, WhatsConnected, WhatsUp, and WS_FTP are registered trademarks of Progress Software Corporation or one of its affiliates or subsidiaries in the U.S. and/or other countries.

Analytics360, AppServer, BusinessEdge, Chef Automate, Chef Compliance, Chef Desktop, Chef Workstation, Corticon Rules, Data Access, DataDirect Autonomous REST Connector, DataDirect Spy, DevCraft, Fiddler, Fiddler Classic, Fiddler Everywhere, Fiddler Jam, FiddlerCap, FiddlerCore, FiddlerScript, Hybrid Data Pipeline, iMail, InstaRelinker, JustAssembly, JustDecompile, JustMock, KendoReact, OpenAccess, PASOE, Pro2, ProDataSet, Progress Results, Progress Software, ProVision, PSE Pro, Push Jobs, SafeSpaceVR, Sitefinity Cloud, Sitefinity CMS, Sitefinity Digital Experience Cloud, Sitefinity Feather, Sitefinity Thunder, SmartBrowser, SmartComponent, SmartDataBrowser, SmartDataObjects, SmartDataView, SmartDialog, SmartFolder, SmartFrame, SmartObjects, SmartPanel, SmartQuery, SmartViewer, SmartWindow, Supermarket, SupportLink, Unite UX, and WebClient are trademarks or service marks of Progress Software Corporation and/or its subsidiaries or affiliates in the U.S. and other countries. Java is a registered trademark of Oracle and/or its affiliates. Any other marks contained herein may be trademarks of their respective owners.

Please refer to the NOTICE.txt or Release Notes – Third-Party Acknowledgements file applicable to a particular Progress product/hosted service offering release for any related required third-party acknowledgements.

Table of Contents

1 Introduction	4
1.1 Document Purpose	4
1.2 Intended Audience	4
2 Kemp Digital Signatures	5
3 Validating the XML Signature	6
4 XML Signature Validation Tools	7
4.1 XMLSec Library	7
4.2 XML Validation using PowerShell	7
Last Updated Date	11

1 Introduction

1.1 Document Purpose

The purpose of this document is to outline how to manually validate the digital signatures of resources provided by Kemp. Resources such as firmware, templates, patches and add-ons have associated XML files that contain the digital signature for the resource.

As of firmware version 7.2.49, you can automatically validate digital signatures easily using the LoadMaster Web User Interface (WUI). Simply enable the **Display Verify Update Option** check box in **System Configuration > Miscellaneous Options > WUI Settings**. This provides an option to upload the XML verification file when updating the LoadMaster software or installing an add-on file on the **System Configuration > System Administration > Update Software** page.

1.2 Intended Audience

This document is intended to guide any LoadMaster administrator or corporate security officer through the options to validate the integrity and authenticity of downloaded Kemp resources.

2 Kemp Digital Signatures

Kemp provides digital signatures as XML signature files (known as detached signatures) as defined by the World Wide Web Consortium (W3C) recommendation for XML Signature Syntax and Processing. Kemp XML signatures use a Kemp certificate issued by a public CA (Certification Authority) to enable validation of authenticity against a known external authority (the public CA).

The Kemp XML signature filenames include the original resource filename as a prefix with the extension .checksum.xml and are provided as separate file downloads when downloading the original Kemp resource.

3 Validating the XML Signature

There are a number of different approaches to validation of detached XML signature files. The XML checksum file is viewable in a text editor and looks something like the following:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<file name="LoadMaster-VLM-7.2.36.0.13835.RELEASE-Microsoft-HyperV.zip">
<checksum>
<sha256>764d88ed0a165a5f43e4b978ff9d4a35ed314b1386779fe213801afd6f05ddfe</sha256>
<md5>819a1c84c13524a326a1937ffdf8e62b</md5>
</checksum>
<Signature xmlns="http://www.w3.org/2000/09/xmldsig#"><SignedInfo><CanonicalizationMethod Algorithm="http://www.w3.org/2001/04/xmldsig-more#rsa-sha256" /><Reference URI=""><Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature" /><Transform Algorithm="http://www.w3.org/TR/2001/REC-xmldsig-core1-20010612#rsa-sha256" /><DigestValue>w6/ICI9nL52xo7T9X3xBituCWqnZ91eu2Dt6hVY+P0
</DigestValue></Reference></SignedInfo><SignatureValue>
msJDR9/M51OHv40JMrX+ZUbJdHE7dTRMLiYNMuuEJ+389fk2oblVlmykoh4mbounouTl1NryuMfC1wvSjpBa6cS6C3flwNqkfg+j6Dv7bUWPe
GagpeOkYfxjI/mAY/35WuDTfOdL26GG/YA7hLBMsL5z6WMJ5ZPhStCnzjgkagsO5AarMiRCDgIWnmfq01DFZi81TxAefbPOxzBMgOMU1W8wFl
KZ/9IaKVW/OD9n6OHhYA==</SignatureValue><KeyInfo><KeyValue><RSAKeyValue><Modulus>
zUcerldV4VKZ+XPtYfy3/HwbSLJu8HS2urs4QUEF41PJY150P/+ePYKpuOcksc2ln45YO5/zrKTcjgxmMxJhrNDDAelPeZjiDlkidj6a79ms
+K1fc1M79mgARNPCLouy21N7d+MNvOlnfBlvRGJZC2CQeeXJhh/q8RdPmVqXliG8zkIIX+wq+kyVSO1ngYsDJZ4iMmjCAQzdT564zK8potl
raFsYnwwt8QS19+UtmQ==</Modulus><Exponent>AQAB</Exponent></RSAKeyValue></KeyValue><X509Data><X509Certificate>
MIIFhTCCBG2gAwIBAgIQJ4OPLMwAnt0LsB1vKLW5STANBgkqhkiG9w0BAQsFADCBKTELMAkGA1UEBhMCVVMxHTAbBgNVBAoTFFN5bWVudGVj
XR3b3JrMUIwQAYDVQQDEz1TeW1hbnRlYyBDYGFzIEV4dGVuZGVkIFZhbG1kYXRpb24gQ29kZSBTaWduaW5nIENBIC0gRzIwHhcNMjYxMjY
c8AgEDEwJVUzEzZMBcGCysGAQQBgjc8AgECEwHzd2FyZTEdMBsGA1UEDxMUUHQjdmF0ZSBPcmddhbm16YXRpb24xEDAOBgNVBAUTBzUwOD
GA1UEBwwITmV3IFlvcmsxHhZAdBgNVBAoMFktFTVAgVGVjaG5vbG9naWVzIEIuYy4xHhZAdBgNVBAMMFktFTVAgVGVjaG5vbG9naWVzIEIuYy4
Upn7E+1h/Lf8fBtIsm7wdLa6uzhBQQXjU8liXnQ//549gqm45ySxzaWfjlg7n/OspNyODPEzEmGsOMMB6U95mOIOWSJ2Pprv2azExeNHMGaj
aABE08Is67LbU3t34w286Wd8GW9EY1kLYJB55ceOGH+rxFO+ZWpeWlBzOQghf7Cr6TJVI7WeBiWlMniIyaOkIBDN1PnrjMrymi0rh+M9uVgO
3xBKX35S2ZAgMBAAGjggGQMIIBjDAuBgNVHREJzAloCMGCCsGAQUFBwQDoBcwFQwTVVMtREVVMQVdBUkUtNTA4NDZmMjA4BgNVHRMEAJAAMA
vc3cuc3ltY2IuY29tL3N3LmNyYDBgBgNVHSAEWTBXMFBwBwBAEDMEwIwYIKwYBBQUHAgEWF2h0dHBzOi8vZC5zeW1jYi5jb20vY3BzMCU
A1UdJQEB/wQMMAAoGCCsGAQUFBwMDMBGA1UdIwQYMBaAFBZm3ko041CnEYyDsWypxqzNwW6bMB0GA1UdDgQWBBS7S9S1QQmGr0PH7cIL+fos8
y9zdy5zeW1jZC5jb20wJwYIKwYBBQUHMAKGG2h0dHA6Ly9zdEuc3ltY2IuY29tL3N3LmNyYDBANBgkqhkiG9w0BAQsFAAOCAQEAX+DH0srrs
s/U+rsy9kowLQJbk/wayWh6pUanFDYg8WIS8sBqBA7kdXMMwA1165M1LkC3mL4ySqRb6wuhUf60FUOFufXuwfaZftBeJ3gOI6iPJUDUVqSTC
YF832WwxV7tliC9ajVbJYz8rrqBrwUL0wim/a3NVBzk7LRTKxC2cfKR9G4ADnJedqJd6Pha4Xn/ktZHWZndlyHPQtexn6d2fhS2DR7ZP2JHK
</X509Certificate></X509Data></KeyInfo></Signature></file>
```

The basic process to validate the XML is as follows:

1. Perform a local checksum on a downloaded Kemp resource (SHA-256).
2. Open the XML signature file and compare the locally generated checksum with the values contained in the XML signature file. To do this, open the XML signature file in a text editor and compare the values contained in the XML signature file under `<checksum><sha256>`. If these values do not match, then the original resource has been altered and should not be trusted.
3. If the checksums match, validate the XML using an XML signature validation tool.

4 XML Signature Validation Tools

A number of tools exist to validate detached XML signatures as provided by Kemp. Kemp make no specific recommendation of which tool to use as the validation and integrity of the tool must be validated by the customer.

4.1 XMLSec Library

The XMLSec Library (<https://www.aleksey.com/xmlsec/>) is a collection of tools that may be used to generate and validate XML signatures. This site provides sources and downloadable binaries for Windows platforms. This tool is available on many Linux environments as the xmlsec1 command.

```
mo@MMUbuntu:~/Downloads$ xmlsec1 --verify 7.1.35.0.13215.RELEASE.20161003-1545-P
ATCH-MT-LMOS.checksum.xml
OK
SignedInfo References (ok/all): 1/1
Manifests References (ok/all): 0/0
mo@MMUbuntu:~/Downloads$
```

4.2 XML Validation using PowerShell

Kemp have created a PowerShell module that can be used to validate detached XML signatures. No warranty or support is provided with this module, which should be validated by your internal security staff before use. The code for the module is included below.

```
Add-Type -AssemblyName System.Security
```

```
# Add SHA-256 per http://stackoverflow.com/questions/30759119/verifying-xml-signature-in-
powershell-with-pem-certificate
```

```
Add-Type @"
```

```
    public class RSAPKCS1SHA256SignatureDescription :
    System.Security.Cryptography.SignatureDescription
```

```

    {
        public RSAPKCS1SHA256SignatureDescription()
        {
            base.KeyAlgorithm = "System.Security.Cryptography.RSACryptoServiceProvider";
            base.DigestAlgorithm = "System.Security.Cryptography.SHA256Managed";
            base.FormatterAlgorithm =
"System.Security.Cryptography.RSAPKCS1SignatureFormatter";
            base.DeformatterAlgorithm =
"System.Security.Cryptography.RSAPKCS1SignatureDeformatter";
        }

        public override System.Security.Cryptography.AsymmetricSignatureDeformatter
CreateDeformatter(System.Security.Cryptography.AsymmetricAlgorithm key)
        {
            System.Security.Cryptography.AsymmetricSignatureDeformatter
asymmetricSignatureDeformatter =
(System.Security.Cryptography.AsymmetricSignatureDeformatter)
            System.Security.Cryptography.CryptoConfig.CreateFromName
(base.DeformatterAlgorithm);
            asymmetricSignatureDeformatter.SetKey(key);
            asymmetricSignatureDeformatter.SetHashAlgorithm("SHA256");
            return asymmetricSignatureDeformatter;
        }
    }
}

'@
$RSAPKCS1SHA256SignatureDescription = New-Object RSAPKCS1SHA256SignatureDescription
[System.Security.Cryptography.CryptoConfig]::AddAlgorithm
($RSAPKCS1SHA256SignatureDescription.GetType(), "http://www.w3.org/2001/04/xmldsig-more#rsa-
sha256")

Function Get-FileName($initialDirectory){

```



```
[System.Reflection.Assembly]::LoadWithPartialName("System.windows.forms") | Out-Null

$OpenFileDialog = New-Object System.Windows.Forms.OpenFileDialog
$OpenFileDialog.initialDirectory = $initialDirectory
$OpenFileDialog.filter = "XML (*.xml)| *.xml"
$OpenFileDialog.ShowDialog() | Out-Null
$OpenFileDialog.filename
}

function ChkCert($cert)
{
    $tmp = [System.IO.Path]::GetTempFileName()
    $cert | Out-File $tmp
    $Certificate = New-Object System.Security.Cryptography.X509Certificates.X509Certificate2($tmp)
    Remove-Item $tmp -Force
    return $Certificate
}

$file = Get-FileName "C:\\"
$xml = New-Object Xml.XmlDocument
$xml.PreserveWhitespace = $true
$xml.Load($file)

$signed = New-Object System.Security.Cryptography.Xml.SignedXml -ArgumentList $xml
$signed.LoadXml($xml.DocumentElement.Signature)
$x509 = $xml.file.Signature.KeyInfo.X509Data.X509Certificate
$cert = ChkCert($x509)
$thumbprint = $cert.Thumbprint
$issuer = $cert.Issuer
$subject = $cert.Subject
```

```
$serial = $cert.SerialNumber

if ($signed.CheckSignature() -eq $true){

    [System.Windows.Forms.MessageBox]::Show("XML Signature validation succeeded`n`nCertificate
Subject Name: $subject`n`nCertificate Issuer: $issuer`n`nCertificate Serial Number:
$serial`n`nCertificate Thumbprint: $thumbprint")

}

else{

    [System.Windows.Forms.MessageBox]::Show("XML Signature validation failed`n`nCertificate
Subject Name: $subject`n`nCertificate Issuer: $issuer`n`nCertificate Serial Number:
$serial`n`nCertificate Thumbprint: $thumbprint")

}
```

Last Updated Date

This document was last updated on 27 July 2023.