WCF servisa un klienta aktivitāšu trasēšana.

Saturs

[1. WCF Servisa un klienta trasēšana 3](#_Toc299464345)

[1.1. Klienta konfigurācijas datne 3](#_Toc299464346)

[1.2. Servisa konfigurācijas datne 4](#_Toc299464347)

[1.3. Žurnalēšana ar Diagnostic.dll 6](#_Toc299464348)

[1.4. Sinhrons servisa izsaukums 7](#_Toc299464349)

[1.5. Asinhrons servisa izsaukums 8](#_Toc299464350)

# WCF Servisa un klienta trasēšana

Šajā piemērā tiks aprakstītas darbības, kas jāveic, lai piesaistītu esošam WCF servisam un klientam aktivitāšu trasēšanas funkcionalitāti izmantojot iebūvētās žurnalēšanas un trasēšanas funkcijas, kā arī papildus ziņojumu žurnalēšanu ar Diagnostic.dll bibliotēku.

Izpildītais risinājuma piemērs uz kā balstīta šī instrukcija atrodams katalogā *„~\Diagnostic\CS WebService\ex01WCFActivity”.*

## Klienta konfigurācijas datne

Modificēt klienta konfigurācijas datni app.config, izmantojot WCF Configuration Editor vai manuāli pievienojot sekojošus atribūtus:

* Sadaļu system.diagnostics kas satur sadaļas sharedListeners (pēc izvēles) un sources. Sadaļā sharedListeners definēt koplietojamos trasēšanas failus norādot sekojošus atribūtus: atrašanās vieta – initializeData, šajā gadījumā lietojuma katalogā, tips – type, identifikators – name, papildus žurnalēšanas iespējas – traceOutputOptions un citi atribūti pēc nepieciešamības;

<system.diagnostics>

<sharedListeners>

<add initializeData="app\_tracelog.svclog"

type="System.Diagnostics.XmlWriterTraceListener, System,

Version=2.0.0.0, Culture=neutral,

PublicKeyToken=b77a5c561934e089"

name="ServiceModelTraceListener"

traceOutputOptions="Timestamp">

<filter type="" />

</add>

</sharedListeners>

* Sadaļā sources definēt ziņojumu klases source ar identifikatoriem name. Klasēm piesaistīt vismaz vienu listeners failu, norādot kādu no koplietojamajiem failiem pēc tā identifikatora name, vai definējot jaunu failu;

<sources>

<source name="System.ServiceModel"

switchValue="Information,ActivityTracing"

propagateActivity="true">

<listeners>

<add name="ServiceModelTraceListener">

<filter type="" />

</add>

</listeners>

</source>

</sources>

* Sadaļā system.diagnostics definēt atribūtu trace ar īpašību autoflush un vērtību true. Kas norāda ka rakstīšana failā tiks veikta uzreiz;

<trace autoflush="true" />

</system.diagnostics>

* Sadaļu system.serviceModel, kas satur sadaļas diagnostic, bindings un client. Sadaļā diagnostic iespējot vai atspējot ziņojumu žurnalēšanas iespējas;

<system.serviceModel>

<diagnostics>

<messageLogging logEntireMessage="true"

logMessagesAtTransportLevel="true" />

</diagnostics>

* Sadaļā bindings uzstādīt servisa piesaistes konfigurāciju;

<bindings>

<basicHttpBinding>

<binding name="BasicHttpBinding\_IService1"

closeTimeout="00:01:00" openTimeout="00:01:00"

receiveTimeout="00:10:00" sendTimeout="00:01:00"

allowCookies="false" bypassProxyOnLocal="false"

hostNameComparisonMode="StrongWildcard"

maxBufferSize="65536" maxBufferPoolSize="524288"

maxReceivedMessageSize="65536"

messageEncoding="Text" textEncoding="utf-8"

transferMode="Buffered" useDefaultWebProxy="true">

<readerQuotas maxDepth="32" maxStringContentLength="8192"

maxArrayLength="16384" maxBytesPerRead="4096"

maxNameTableCharCount="16384" />

<security mode="None">

<transport clientCredentialType="None"

proxyCredentialType="None"

realm="" />

<message clientCredentialType="UserName"

algorithmSuite="Default" />

</security>

</binding>

</basicHttpBinding>

</bindings>

* Sadaļā client piesaistīt servisu - adress un norādīt piesaistes tipu - binding un tās konfigurāciju – bindingConfiguration.

<client>

<endpoint address="http://localhost:64775/Service1.svc"

binding="basicHttpBinding"

bindingConfiguration="BasicHttpBinding\_IService1"

contract="ServiceRef.IService1"

name="BasicHttpBinding\_IService1" />

</client>

</system.serviceModel>

## Servisa konfigurācijas datne

Modificēt servisa konfigurācijas datni web.config, izmantojot WCF Configuration Editor vai manuāli pievienojot sekojošus atribūtus:

* Sadaļu system.diagnostics kas satur sadaļas sharedListeners (pēc izvēles) un sources. Sadaļā sharedListeners definēt koplietojamos trasēšanas failus norādot sekojošus atribūtus: atrašanās vieta – initializeData, šajā gadījumā lietojuma katalogā, tips – type, identifikators – name, papildus žurnalēšanas iespējas – traceOutputOptions un citi atribūti pēc nepieciešamības;

<system.diagnostics>

<sharedListeners>

<add initializeData="Web\_tracelog.svclog"

type="System.Diagnostics.XmlWriterTraceListener, System,

Version=2.0.0.0, Culture=neutral,

PublicKeyToken=b77a5c561934e089"

name="ServiceModelTraceListener"

traceOutputOptions="Timestamp">

<filter type="" />

</add>

</sharedListeners>

* Sadaļā sources definēt ziņojumu klases source ar identifikatoriem name. Klasēm piesaistīt vismaz vienu listeners failu, norādot kādu no koplietojamajiem failiem pēc tā identifikatora name, vai definējot jaunu failu;

<sources>

<source name="System.ServiceModel"

switchValue="Information,ActivityTracing"

propagateActivity="true">

<listeners>

<add name="ServiceModelTraceListener">

<filter type="" />

</add>

</listeners>

</source>

</sources>

Sadaļā system.diagnostics definēt atribūtu trace ar īpašību autoflush un vērtību true. Kas norāda ka rakstīšana failā tiks veikta uzreiz;

<trace autoflush="true" />

</system.diagnostics>

* Sadaļu system.serviceModel, kas satur sadaļas diagnostic, services un behaviors. Sadaļā diagnostic iespējot vai atspējot ziņojumu žurnalēšanas iespējas;

<system.serviceModel>

<diagnostics>

<messageLogging logEntireMessage="true"

logMessagesAtTransportLevel="true" />

</diagnostics>

* Sadaļā services uzstādīt servisa konfigurāciju;

<services>

<service behaviorConfiguration="WcfService.Service1Behavior"

name="WcfService.Service1">

<endpoint address="" binding="basicHttpBinding"

bindingConfiguration=""

contract="WcfService.IService1">

<identity>

<dns value="localhost" />

</identity>

</endpoint>

<endpoint address="mex" binding="mexHttpBinding"

contract="IMetadataExchange" />

</service>

</services>

* Sadaļā behaviors uzstādīt servisa konfigurāciju.

<behaviors>

<serviceBehaviors>

<behavior name="WcfService.Service1Behavior">

<serviceMetadata httpGetEnabled="true"/>

<serviceDebug includeExceptionDetailInFaults="false"/>

</behavior>

</serviceBehaviors>

</behaviors>

</system.serviceModel>

## Žurnalēšana ar Diagnostic.dll

Trasējamības uzlabošanai tiek piesaistīta papildus žurnalēšana gan servisa, gan klienta pusē izmatojot LogUtility;

* Papildināt konfigurācijas failus web.config un app.config;

<!--for Diagnostic.dll logUtility-->

<source name ="Activity" switchValue="All">

<listeners>

<add name="ServiceModelTraceListener"></add>

</listeners>

</source>

* Pievienot klienta un servisa risinājumiem Diagnostic.dll bibliotēku un atbilstošas vārdtelpas;
* Izveidot žurnalēšanas metodes nepieciešamajās vietās gan servisa gan klienta pusē;

LogUtility logWriter = new LogUtility("WCFService");

public string GetData(int value)

{

logWriter.Write("Received value: " + value.ToString(), "Activity");

System.Threading.Thread.Sleep(value);

return string.Format("You entered: {0}", value);

}

logWriter.Write("Entered: " + i.ToString(), "Activity");

rez = c.GetData(i);

logWriter.Write("Response: " + rez, "Activity");

OutputTextBox.Text = rez;

## Sinhrons servisa izsaukums

Sinhroni servisa izsaukumi var tikt automātiski apstrādāti bez papildus kodēšanas. Bet, lai nodrošinātu korektu aktivitāšu atspoguļojumu tiek izmantota LogActivity klases funkcionalitāte (Ilustrācija 1‑1);

private void SyncGetDataButton\_Click(object sender, EventArgs e)

{

int i;

string rez;

LogActivity la = LogActivity.CreateBoundedActivity();

la.Start("SyncCall", "FromClient");

using (ServiceRef.Service1Client c = new ServiceRef.Service1Client())

{

if (!int.TryParse(InputTextBox.Text, out i))

{

OutputTextBox.Text = "Ievadīt skaitli!";

}

else

{

logWriter.Write("Entered: " + i.ToString(), "Activity");

rez = c.GetData(i);

logWriter.Write("Response: " + rez, "Activity");

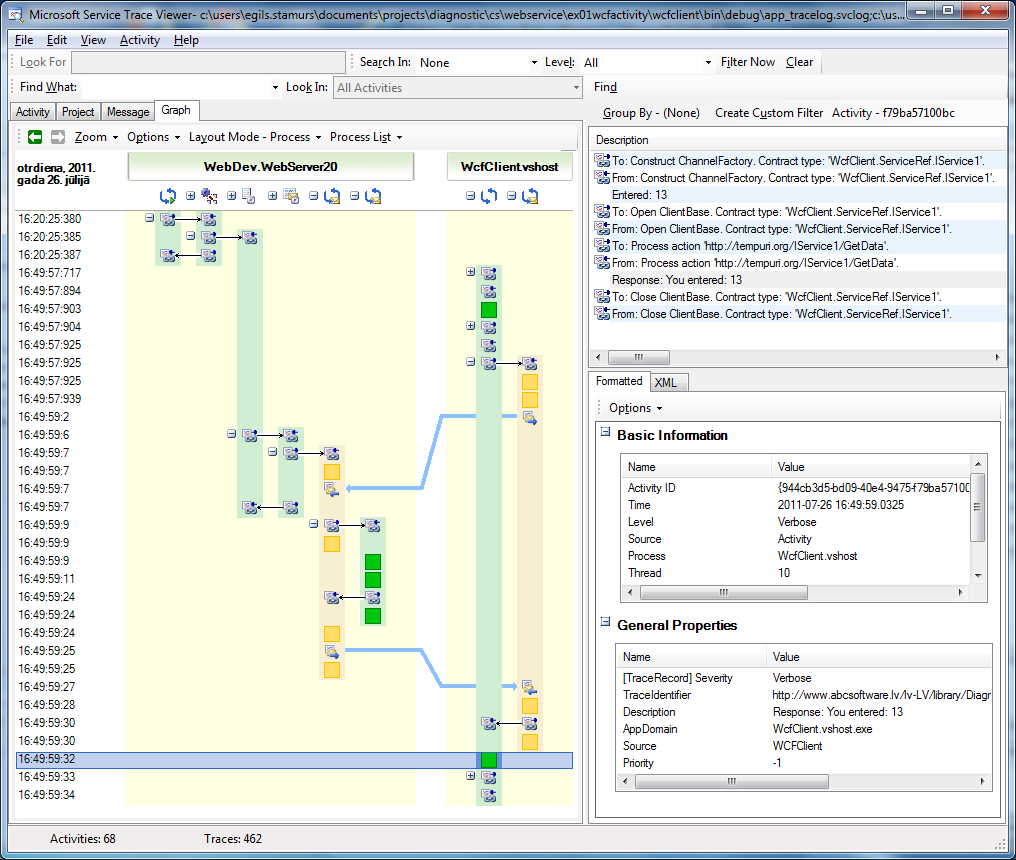
OutputTextBox.Text = rez;

}

}

la.Stop();

}



Ilustrācija ‑ Sinhrona izsaukuma rezultāts

## Asinhrons servisa izsaukums

Asinhrona servisa izsaukuma gadījumā nepieciešams izmantot LogActivity klases funkcionalitāti, lai nodrošinātu korektas ziņojumu identifikatoru vērtības izsaucot servisu un atiežoties no tā (Ilustrācija 1‑2). Metode AsyncGetDataButton\_Click nodrošina asinhronu servisa izsaukumu:

private void AsyncGetDataButton\_Click(object sender, EventArgs e)

{

int i;

LogActivity l = LogActivity.CreateAsyncActivity();

using (LogActivity la = LogActivity.CreateBoundedActivity(l.Id))

{

la.Start("AsyncCall", "FromClient");

if (!int.TryParse(InputTextBox.Text, out i))

{

OutputTextBox.Text = "Ievadīt skaitli!";

}

else

{

logWriter.Write("Async Entered: " + i.ToString(), "Activity");

ServiceRef.Service1Client c = new ServiceRef.Service1Client();

l.Suspend();

c.GetDataAsync(Convert.ToInt32(this.InputTextBox.Text), l);

c.GetDataCompleted += new EventHandler<ServiceRef.

GetDataCompletedEventArgs>(c\_GetDataCompleted);

}

}

}

Metode c\_GetDataCompleted nodrošina datu izvadi pēc izsaukuma izpildīšanas;

private void c\_GetDataCompleted(object sender, ServiceRef.GetDataCompletedEventArgs e)

{

LogActivity l = e.UserState as LogActivity;

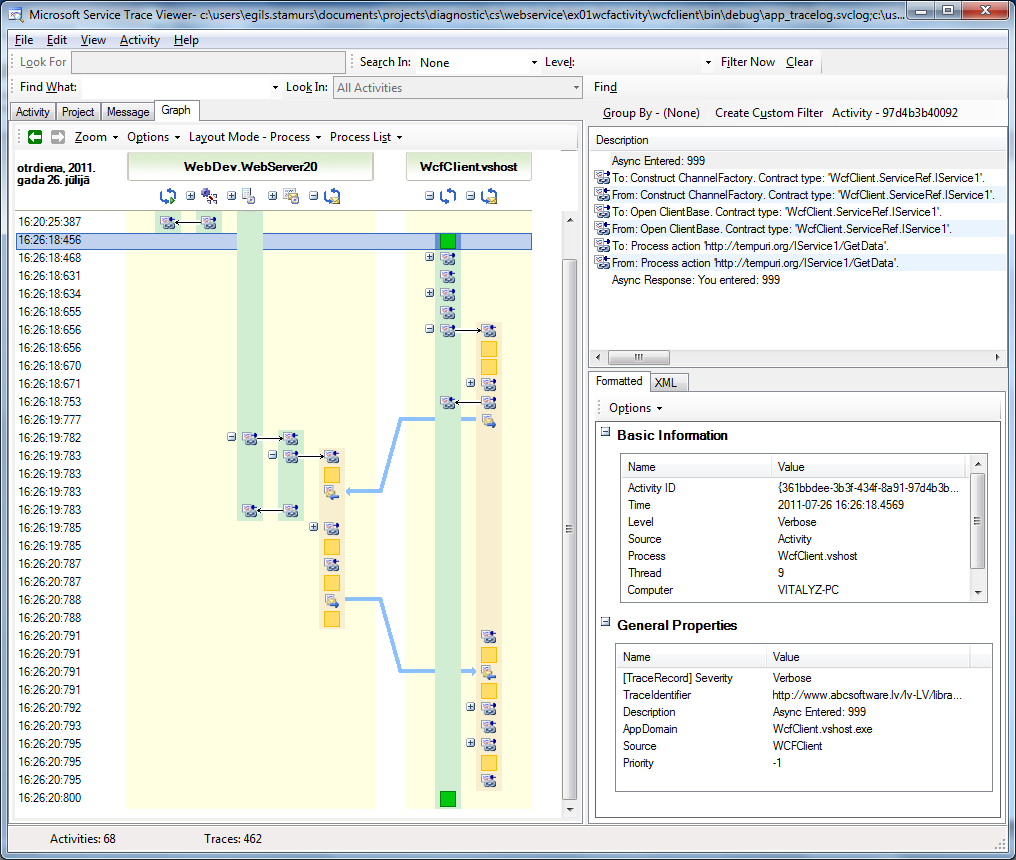
l.Resume();

this.OutputTextBox.Text = e.Result;

logWriter.Write("Async Response: " + e.Result, "Activity", -1, -1, System.Diagnostics.TraceEventType.Verbose, l.Id);

l.Stop();

}



Ilustrācija ‑ Asinhrona izsaukuma rezultāts