add following to web.config file in the configuration tag:

<system.diagnostics>

<sources>

<source name="System.ServiceModel" switchValue="Information, ActivityTracing, Critical, Error" propagateActivity="true">

<listeners>

<add name="traceListener" type="System.Diagnostics.XmlWriterTraceListener" initializeData="D:\log\Traces.svclog" />

</listeners>

</source>

<source name="CardSpace">

<listeners>

<add name="xml" />

</listeners>

</source>

<source name="System.IO.Log">

<listeners>

<add name="xml" />

</listeners>

</source>

<source name="System.Runtime.Serialization" switchValue="Information, ActivityTracing, Critical, Error">

<listeners>

<add name="xml" />

</listeners>

</source>

<source name="System.IdentityModel">

<listeners>

<add name="xml" />

</listeners>

</source>

<source name="UserTraceSource" switchValue="Information, ActivityTracing, Critical, Error">

<listeners>

<add name="xml" type="System.Diagnostics.XmlWriterTraceListener" initializeData="D:\log\Traces.svclog" />

</listeners>

</source>

<source name="System.ServiceModel.MessageLogging">

<listeners>

<add name="messages"

type="System.Diagnostics.XmlWriterTraceListener"

initializeData="D:\log\Traces.svclog" />

</listeners>

</source>

</sources>

<trace autoflush="true" />

<sharedListeners>

<add name="xml" type="System.Diagnostics.XmlWriterTraceListener" initializeData="D:\log\Traces.svclog" />

</sharedListeners>

</system.diagnostics>

and add this to <system.serviceModel> :

<diagnostics>

<messageLogging

logEntireMessage="true"

logMalformedMessages="false"

logMessagesAtServiceLevel="true"

logMessagesAtTransportLevel="false"

maxMessagesToLog="30000"

maxSizeOfMessageToLog="20000"/>

</diagnostics>

create manually the file: d:\log\Traces.svclog

test service

open file with Microsoft Service Trace Viewer

<https://docs.microsoft.com/en-us/dotnet/framework/wcf/diagnostics/tracing/configuring-tracing>

<https://docs.microsoft.com/en-us/dotnet/framework/wcf/diagnostics/configuring-message-logging>

info:

Tracing is not enabled by default. To activate tracing, you must create a trace listener and set a trace level other than "Off" for the selected trace source in configuration; otherwise, WCF does not generate any traces. If you do not specify a listener, tracing is automatically disabled. If a listener is defined, but no level is specified, the level is set to "Off" by default, which means that no trace is emitted.

The tracing level is controlled by the switchValue setting of the trace source

| Trace Level | Nature of the Tracked Events | Content of the Tracked Events | Tracked Events | User Target |
| --- | --- | --- | --- | --- |
| Off | N/A | N/A | No traces are emitted. | N/A |
| Critical | "Negative" events: events that indicate an unexpected processing or an error condition. |  | Unhandled exceptions including the following are logged:  - OutOfMemoryException - ThreadAbortException (the CLR invokes any ThreadAbortExceptionHandler) - StackOverflowException (cannot be caught) - ConfigurationErrorsException - SEHException - Application start errors - Failfast events - System hangs - Poison messages: message traces that cause the application to fail. | Administrators  Application developers |
| Error | "Negative" events: events that indicate an unexpected processing or an error condition. | Unexpected processing has happened. The application was not able to perform a task as expected. However, the application is still up and running. | All exceptions are logged. | Administrators  Application developers |
| Warning | "Negative" events: events that indicate an unexpected processing or an error condition. | A possible problem has occurred or may occur, but the application still functions correctly. However, it may not continue to work properly. | - The application is receiving more requests than its throttling settings allow. - The receiving queue is near its maximum configured capacity. - Timeout has exceeded. - Credentials are rejected. | Administrators  Application developers |
| Information | "Positive" events: events that mark successful milestones | Important and successful milestones of application execution, regardless of whether the application is working properly or not. | In general, messages helpful for monitoring and diagnosing system status, measuring performance or profiling are generated. You can use such information for capacity planning and performance management:  - Channels are created. - Endpoint listeners are created. - Message enters/leaves transport. - Security token is retrieved. - Configuration setting is read. | Administrators  Application developers  Product developers. |
| Verbose | "Positive" events: events that mark successful milestones. | Low level events for both user code and servicing are emitted. | In general, you can use this level for debugging or application optimization.  - Understood message header. | Administrators  Application developers  Product developers. |
| ActivityTracing |  | Flow events between processing activities and components. | This level allows administrators and developers to correlate applications in the same application domain:  - Traces for activity boundaries, such as start/stop. - Traces for transfers. | All |
| All |  | Application may function properly. All events are emitted. | All previous events. | All |

The activityTracing value specified for the switchValue attribute is used to enable activity tracing, which emits traces for activity boundaries and transfers within endpoints.

When you use certain extensibility features in WCF, you might get a [NullReferenceException](https://docs.microsoft.com/en-us/dotnet/api/system.nullreferenceexception) when activity tracing is enabled. To fix this problem, check your application's configuration file and ensure that the switchValue attribute for your trace source is not set to activityTracing.

The propagateActivity attribute indicates whether the activity should be propagated to other endpoints that participate in the message exchange. By setting this value to true, you can take trace files generated by any two endpoints and observe how a set of traces on one endpoint flowed to a set of traces on another endpoint.