Service Profiler

for IS webMethods

wrightia

Why is profiling a business need?

These are times where changes happen very fast pushed by more aggressive competition. In order to achieve more customer sales or decrease operation costs most companies are investing in more agile processes, applications, and technology architectures.

Flexibility of information technologies allows organizations to continually transform and thrive in a dynamic environment, but comes at a cost. The IT systems become more complex, harder to maintain and operate.

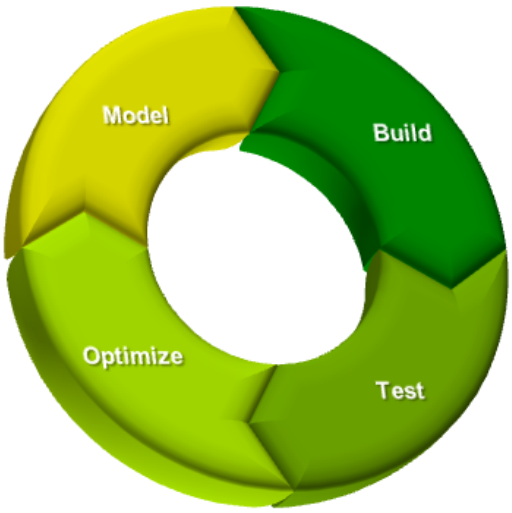
Those systems responsiveness becomes even more important and may have a direct impact on business revenue.

Building systems for performance is mainly an architectural and design task but the truth is that it is often overlooked, becoming only visible too late in the development life-cycle. That is when a quick and effective audit of the current implementation, with pragmatic optimization hints both for short term quick-wins as well as longer term performance improvements, becomes vital.

In order to help organizations achieve a better understanding of the implemented services over the webMethods platform we have developed the Service Profiler, this helps you tracking down performance bottlenecks or dead code, with close to zero runtime overhead on development, Q&A and production environments. ❖

What are the benefits for each phase in the application cycle ?

From "Model", through "Build", "Test "and "Optimize" phases, all need to base themselves on some solution profiling in order to cope with specifications. These may incorporate concerns about performance, reliability, flexibility, management, etc.; and that may have to consider raw information such as data volumes, day time schedules, number of users, component distribution, etc.



Model — the architecture lays out the adequate solution considering the involved hardware and software components that match the problem and its environment, profiling is materialized by outlining the application/solution.

Build— the actual building of the solution takes place here, profiling activity decreases because most of the behavioural intentions have already been established by the Model phase.

Test— after "Build" phase is completed, it is time to test the solution for bugs, error handling and against the established specifications. During this phase profiling helps identifying what is causing differences on behaviour between what is expected and the witnessed results. Another advantage of the Service Profiler is the ability to identify dead code that could be removed or rearranged.

Optimize— Once in a production environment, it's important to have a proactive role in the identification of performance bottlenecks since real live situations are hard to replicate in a Q&A environment. ❖

"The capability we now possess with the Service Profiler to quickly and easily identify all aspects under our solution that could negatively influence its performance allows us to be more efficient in it's detection and correction, hence saving us from such efforts in more advanced stages of the development cycle."

Dr. João Pedro Silva
IT - Director, Vodafone Portugal

Key features

- **Easy Install & Uninstall -** the installation and uninstall are simple and straight foreward. The tool can be made completely inactive without uninstalling.
- **Non-intrusive -** no changes need to be made to either the Integration Server or the Services.
- **Small impact on Integration Server resources/performance -** the tool uses a very small amount of memory to gather data and negligible CPU time.
- **Any Service type -** the tool works with all the services available in the webMethods platform (Flow Service, Java Service, Adapter Service, C/C++ Service, XSLT Service, Webservice).
- **Fine-grained timing data -** the values collected for the Service execution goes beyond the simple gathering of timestamps, it actualy records the time spent in the CPU.
- **Simple set of operations -** the operations are very intuitive and straighforward (Start, Stop, Analyze, Take Snapshot, Export Data).
- **Open to external Analysis Tools -** snapshots can be exported to a variety of standard exchange file formats or be accessed via a set of public services.
- **Administration pages -** the tool's administration and configuration settings are accessed through a browser.
- **Dashboard presentation -** a set of graphics ease the reading and drilling of statistical data gathered by the tool.

Different environments different needs

"Time is Money" is a fairly common saying in the business world. While the usage of the Service Profiler *per se* does not imply that the services run faster, it allows saving time in tasks that are themselves dedicated to overcome factual or potential points where time could be at waste. These tasks are usually of diagnostics, testing and monitoring.

While in a Development environment, profiling can help fine tuning the specific artefacts to comply to performance requirements to guarantee that they will not be a bottleneck.

Once in a Testing environment, with data loads and runtime environment closer to production specifications, Service Profiler allows testers to more easily measure execution times against requirements or SLAs established by business rules. When unacceptable execution times are then witnessed, the breakdown reporting of a Service Profile cuts short the diagnosis and identification of the performance culprit or culprits.

In a Production environment the Service Profiler can be a valuable asset for reactive monitoring, enabling a quick and effective live performance analysis to be carried out, without changing a line of code or restarting the Integration Server. Once installed you can keep the Service Profiler dormant, without any impact on your Production system, until you actually need it.❖

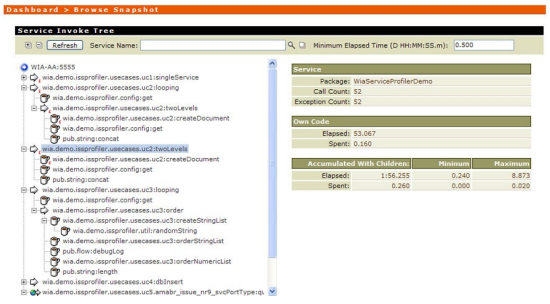
The Service Profiler was designed to help you fine tune your applications

The Service Profiler's most fundamental job is to collect raw execution data about Integration Services. The nature of this data includes: execution elapsed times, execution CPU times, number of times called, number of exceptions raised, and execution path.

Around this core functionality of gathering raw data other functionalities emerge on areas of administration/configuration, data analysis (dashboards) and extensibility. These functional areas respectively aim to control tool behaviour, translate the raw data into meaningful information and to allow processing of collected data by external tools.

Start and Stop — command whether the Service profiler is actively collecting data or not.

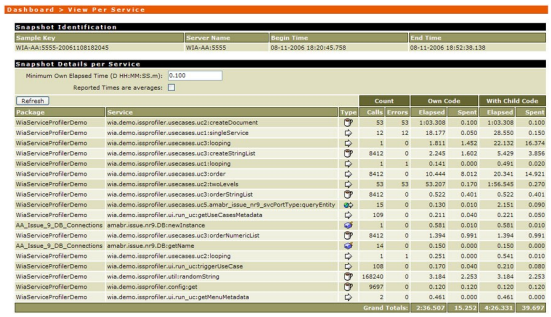
Browse Snapshot— is a dashboard analysis tool that shows the execution paths of all collected services as nodes in an execution tree. Each node can be selected to view its detailed information.



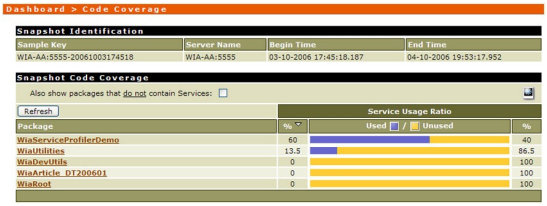
Browse Running Services — regular snapshots return data about the services that have already returned or completed. When there is one or more Services taking a long time to complete it is useful to be able to take a peek at what is happening. This functionality lists the threads that are currently running services with execution paths spanning from them as trees similar to those viewed by Browse Snapshot. The currently executing Service and those currently on the call-stack are identified. Detail information about each service in the tree can be viewed by selecting its corresponding node.

Freeze Snapshot — information processing is based on snapshots taken on request. Each dashboard and export requests a new snapshot every time they are called. However, analysis of the data with multiple analysis tools and export is useful without the noise of having it refreshed every time it is viewed. For this purpose the snapshot can be frozen. And in this state the data collecting is kept running but every time a snapshot is requested a frozen one is returned. When the data collecting is Stopped, a snapshot is frozen so that the tool can be used to work on the latest known snapshot.

View Per Service — on this analysis tool the execution data is presented in a tabular form, accumulated per service, independently of where the service was called from or which service(s) it in turn called. The table can be sorted by any column, some filters can be defined to exclude unimportant data, and a drilldown on a selected service can be made into the Browse Snapshot dashboard, highlighting the nodes for that Service.



Code Coverage — it shows the ratio of services executed against the number Services defined in each Integration Server Package. It can help to determine if the Service distribution into packages is balanced with their execution profile.



Export — a data snapshot can be exported as a file in CSV or XML format.

Package Exclusion Patterns — on an Integration Server there are packages from several sources including webMethods and 3rd party. Because all running services contribute to the overall performance of the server, none of them is discarded of filtered out from data gathering. However, not all of them need to be shown in the dashboard pages, mainly because no Refactoring can be attempted over them. This filtering is defined with Regular Expressions over the package name. ❖

Available on the following Operating Systems

- Windows 2000 & XP
- Windows Server 2003
- Sun Solaris
- HP-UX
- Linux

wrightia

Wrightia is an integration company and a webMethods partner, with customers around Europe.

For more information about Service Profiler, please contact our sales oficce using:

sales@wrightia.com
+351 214 259 827