Article
Murray Oldfield · Nov 14, 2019 6m read

Monitoring InterSystems IRIS Using Built-in REST API

Released with no formal announcement in IRIS preview release 2019.4 is the /api/monitor service exposing IRIS metrics in ***Prometheus*** format. Big news for anyone wanting to use IRIS metrics as part of their monitoring and alerting solution. The API is a component of the new IRIS *System Alerting and Monitoring (SAM)* solution that will be released in an upcoming version of IRIS.

> However, you do not have to wait for SAM to start planning and trialling this API to monitor your IRIS instances. In future posts, I will dig deeper into the metrics available and what they mean and provide example interactive dashboards. But first, let me start with some background and a few questions and answers.

IRIS (and Caché) is always collecting dozens of metrics about itself and the platform it is running on. There have always been multiple ways to collect these metrics to monitor Caché and IRIS. I have found that few installations use IRIS and Caché built-in solutions. For example, History Monitor has been available for a long time as a historical database of performance and system usage metrics. However, there was no obvious way to surface these metrics and instrument systems in real-time.

IRIS platform solutions (along with the rest of the world) are moving from single monolithic applications running on a few on-premises instances to distributed solutions deployed 'anywhere'. For many use cases existing IRIS monitoring options do not fit these new paradigms. Rather than completely reinvent the wheel InterSystems looked to popular and proven current Open Source solutions for monitoring and alerting.

Prometheus?

Prometheus is a well known and widely deployed open source monitoring system based on proven technology. It has a wide variety of plugins. It is designed to work well within the cloud environment, but also is just as useful for on-premises. Plugins include operating systems, web servers such as Apache and many other applications. Prometheus is often used with a front end client, for example, *Grafana*, which provides a great UI/UX experience that is extremely customisable.

Grafana?

Grafana is also open source. As this series of posts progresses, I will provide sample templates of monitoring dashboards for common scenarios. You can use the samples as a base to design dashboards for what you care about. The real power comes when you combine IRIS metrics in context with metrics from your whole solution stack. From the platform components, operating system, IRIS and especially when you add instrumentation from your applications.

Haven't I seen this before?

Monitoring IRIS and Caché with Prometheus and Grafana is not new. I have been using these applications for several years to monitor my development and test systems. If you search the Developer Community for "Prometheus", you will find other posts (for example, some excellent posts by Mikhail Khomenko) that show how to expose Caché metrics for use by Prometheus.

> The difference now is that the /api/monitor API is included and enabled by default. There is no requirement to code your own classes to expose metrics.

Prometheus Primer

Here is a quick orientation to Prometheus and some terminology. I want you to see the high level and to lay some groundwork and open the door to how you think of visualising or consuming the metrics provided by IRIS or other sources.

Prometheus works by *scraping* or pulling time series data exposed from applications as HTTP endpoints (APIs such as IRIS /api/monitor). *Exporters* and client libraries exist for many languages, frameworks, and open-source applications — for example, for web servers like Apache, operating systems, docker, Kubernetes, databases, and now IRIS.

Exporters are used to instrument applications and services and to expose relevant metrics on an endpoint for scraping. Standard components such as web servers, databases, and the like - are supported by core exporters. Many other exporters are available open-source from the Prometheus community.

Prometheus Terminology

A few key terms are useful to know:

- **Targets** are where the services are that you care about, like a host or application or services like Apache or IRIS or your own application.
- Prometheus **scrapes** targets over HTTP collecting metrics as time-series data.
- **Time-series data** is exposed by applications, for example, IRIS or via exporters.
- **Exporters** are available for things you don't control like Linux kernel metrics.
- The resulting time-series data is stored locally on the Prometheus server in a database **.
- The time-series database can be queried using an optimised **query language (PromQL)**. For example, to create alerts or by client applications such as Grafana, to display the metrics in a dashboard.

  ** **Spoiler Alert;** For security, scaling, high availability and some other operational efficiency reasons, for the new SAM solution the database used for Prometheus time-series data is IRIS! However, access to the Prometheus database -- on IRIS -- is transparent, and applications such as Grafana do not know or care.

Prometheus Data Model

Metrics returned by the API are in Prometheus format. Prometheus uses a simple text-based metrics format with one metric per line, the format is;

```
<identifier> [ (time n, value n), ....]
```

Metrics can have labels as (key, value) pairs. Labels are a powerful way to filter metrics as dimensions. As an example, examine a single metric returned for IRIS /api/monitor. In this case journal free space:

```
iris_jrn_free_space{id="WIJ",dir="/fast/wij/"} 401562.83
```

The identifier tells you what the metric is and where it came from:

```
iris_jrn_free_space
```

Multiple labels can be used to decorate the metrics, and then used to filter and query. In this example, you can see the WIJ and the directory where the WIJ is stored:

```
id="WIJ",dir="/fast/wij/"
```

And a value: 401562.83 (MB).

What IRIS metrics are available?

The preview documentation has a list of metrics. However, be aware there may be changes. You can also simply query the /api/monitor/metrics endpoint and see the list. I use Postman which I will demonstrate in the next community post.

What should I monitor?

Keep these points in mind as you think about how you will monitor your systems and applications.

- When you can, instrument key metrics that affect users.
    - Users don't care that one of your machines is short of CPU.
    - Users care if the service is slow or having errors.
    - For your primary dashboards focus on high-level metrics that directly impact users.
- For your dashboards avoid a wall of graphs.
    - Humans can't deal with too much data at once.
    - For example, have a dashboard per service.
- Think about services, not machines.
    - Once you have isolated a problem to one service, then you can drill down and see if one machine is the problem.

References

Documentation and downloads for: Prometheus and Grafana

I presented a pre-release overview of SAM (including Prometheus and Grafana) at **InterSystems Global Summit 2019** you can find the link at InterSystems learning services. If the direct link does not work go to the InterSystems learning services web site and search for: "System Alerting and Monitoring Made Easy"

Search here on the community for "Prometheus" and "Grafana".

#API #Best Practices #Dashboards #DevOps #InterSystems Business Solutions and Architectures #Monitoring #Open Source #System Administration #InterSystems IRIS #InterSystems IRIS for Health

---
