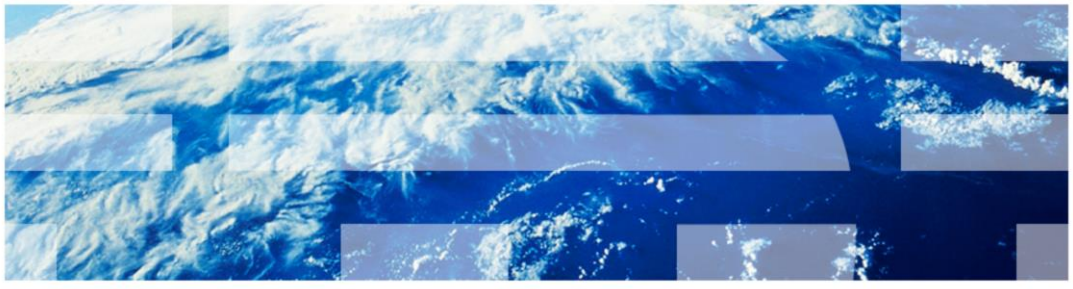
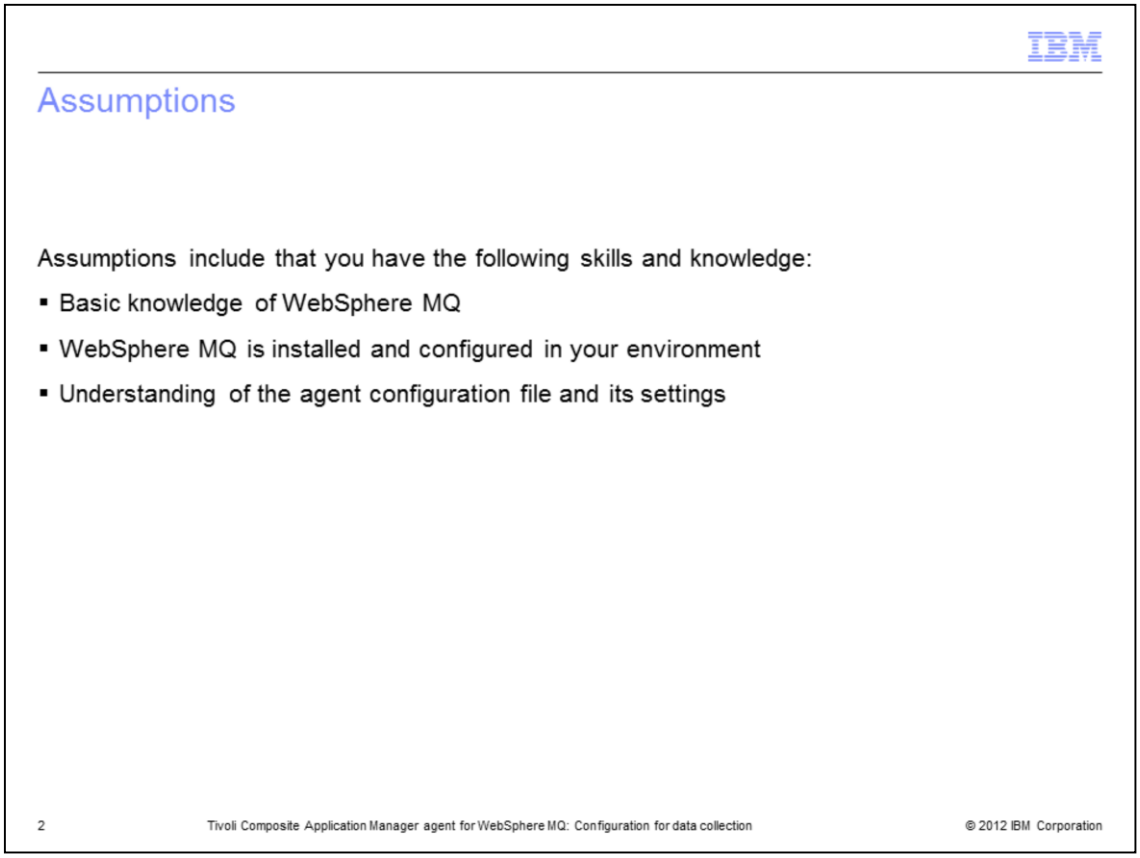
IBM Tivoli Composite Application Manager for Applications V6.2.4

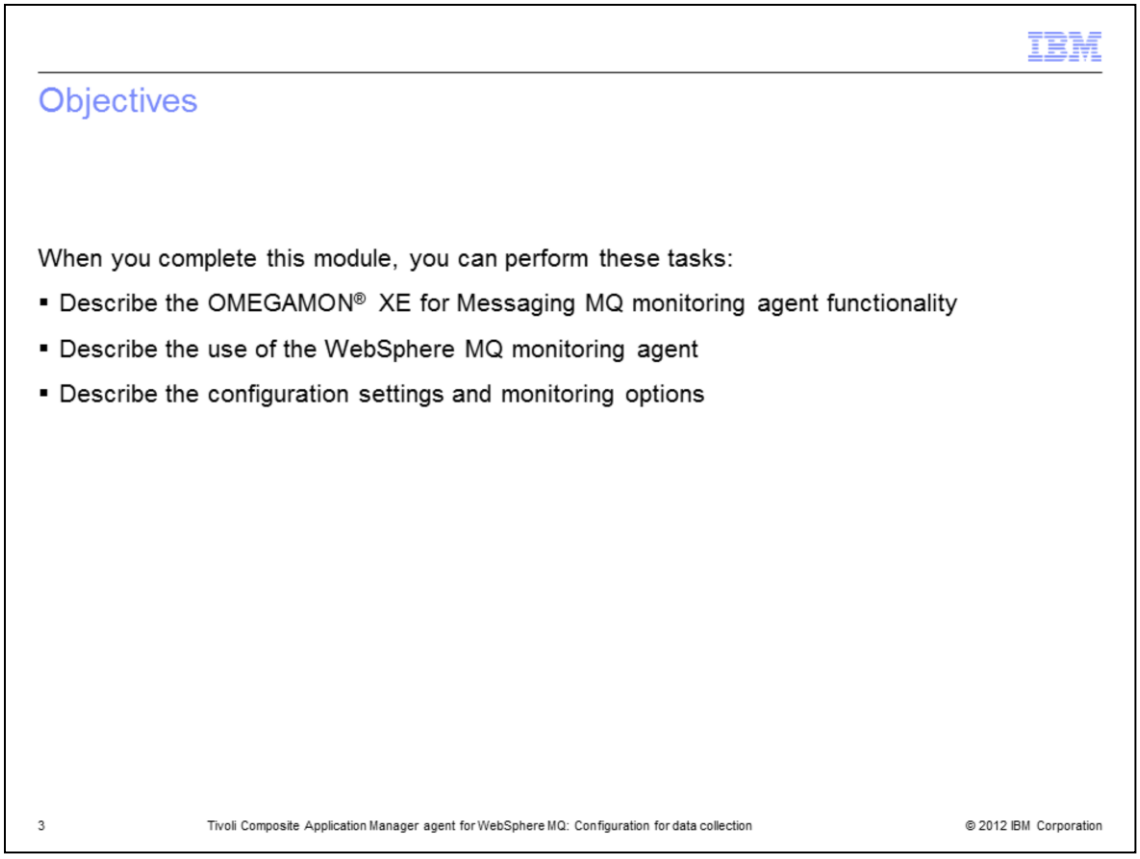
Tivoli Composite Application Manager agent for WebSphere MQ: Configuration for data collection



This presentation provides an overview of what the IBM Tivoli® Composite Application agent for WebSphere® MQ is and how to configure it to collect statistics from WebSphere MQ objects.



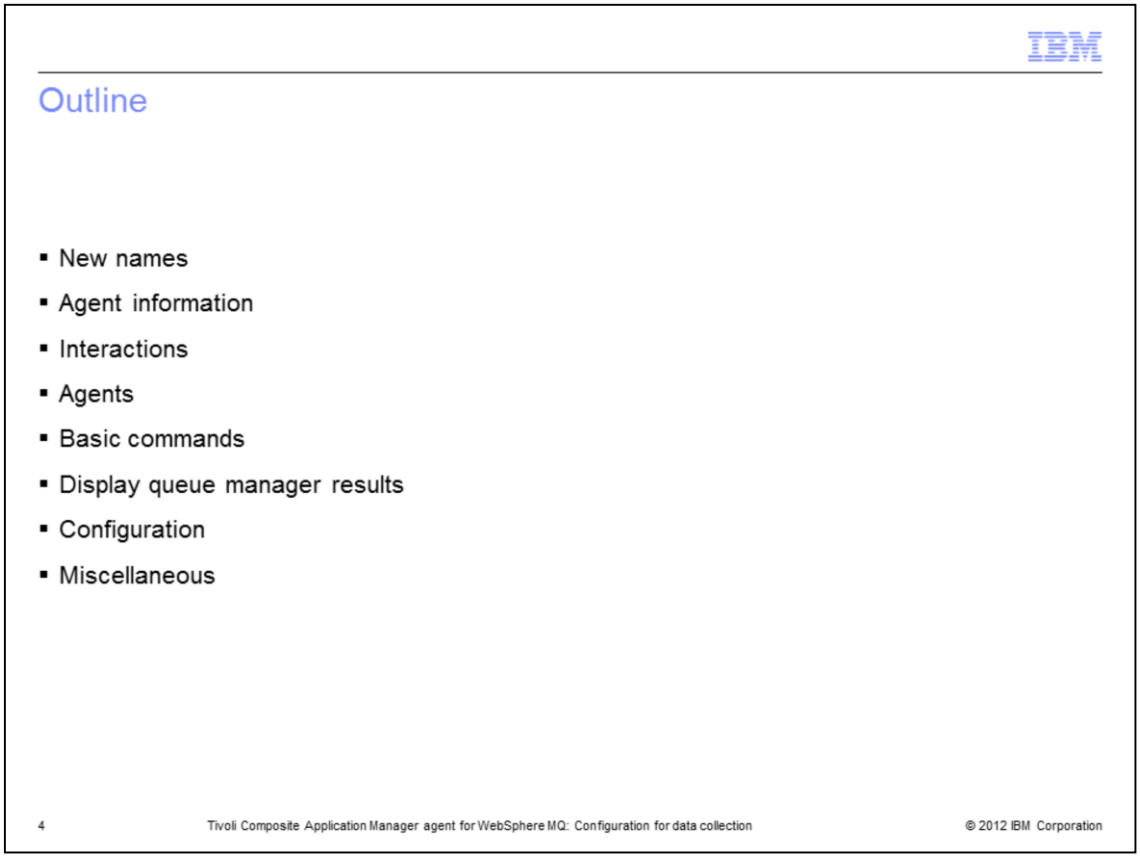
For this module, assumptions include that you have basic knowledge of WebSphere MQ, and have WebSphere MQ installed and configured in your environment.



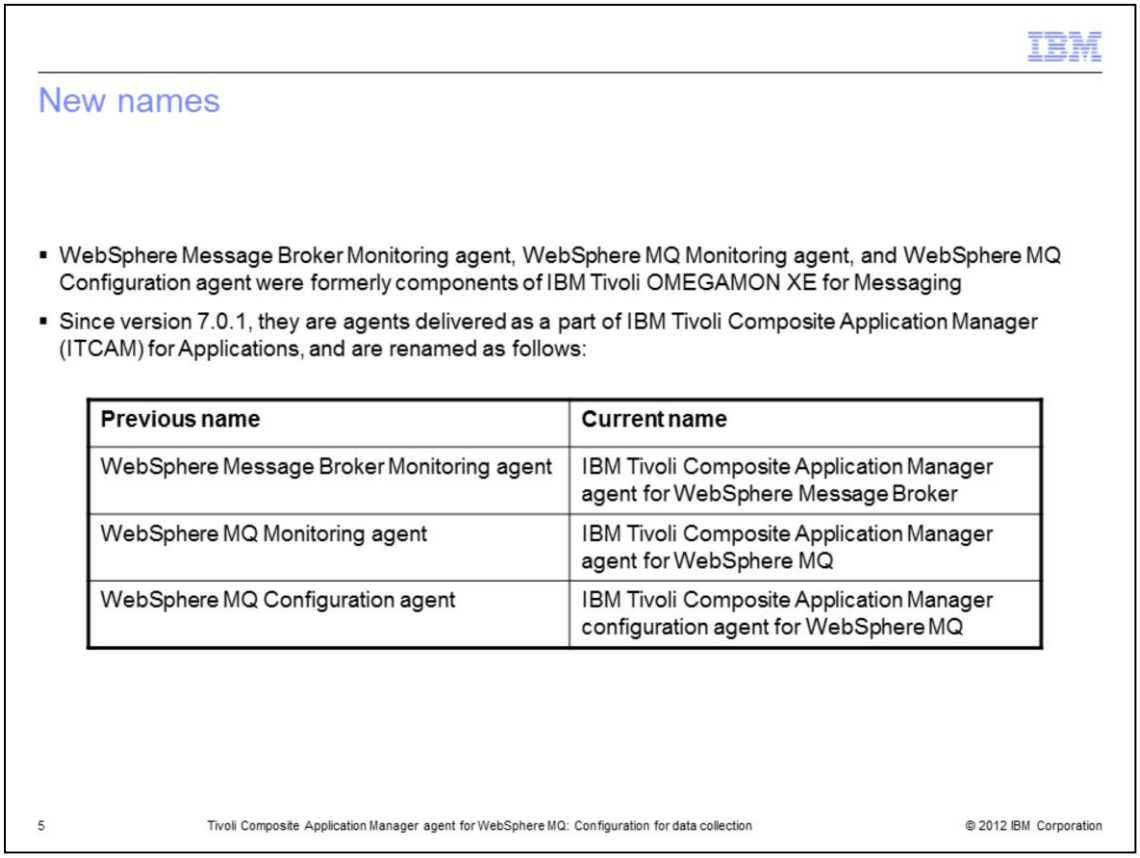
Objectives.

When you complete this module, you can perform these tasks:

- Describe the OMEGAMON XE for Messaging MQ monitoring agent functionality
- Describe the use of the MQ monitoring agent
- Describe the configuration settings and monitoring options

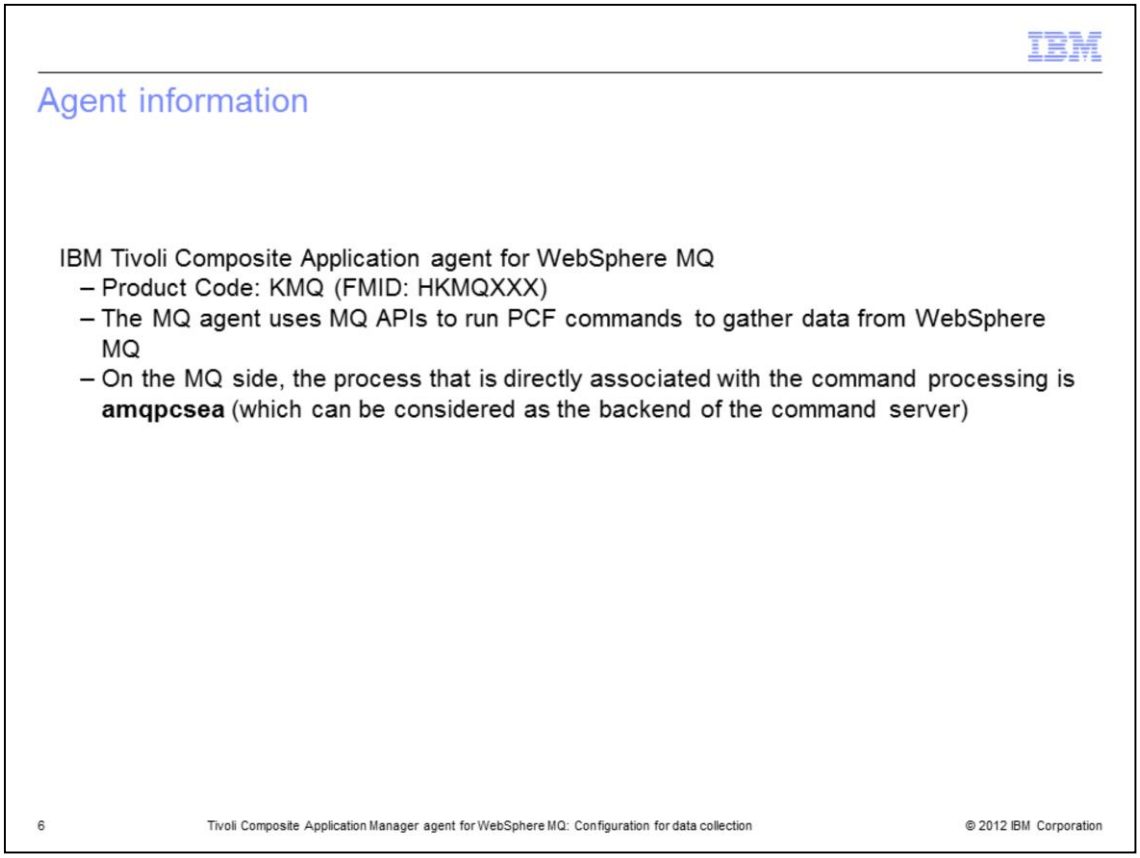


This module outlines the new product names, agent information, interactions, agents, basic commands, display queue manager output, configuration, and other information.



IBM Tivoli Composite Application agent for WebSphere MQ is the new name for the MQ agent that was contained in the OMEGAMON XE for Messaging bundle.

Now, the agent is part of the IBM Tivoli Composite Application for Applications suite, and its latest version is 7.0.1.

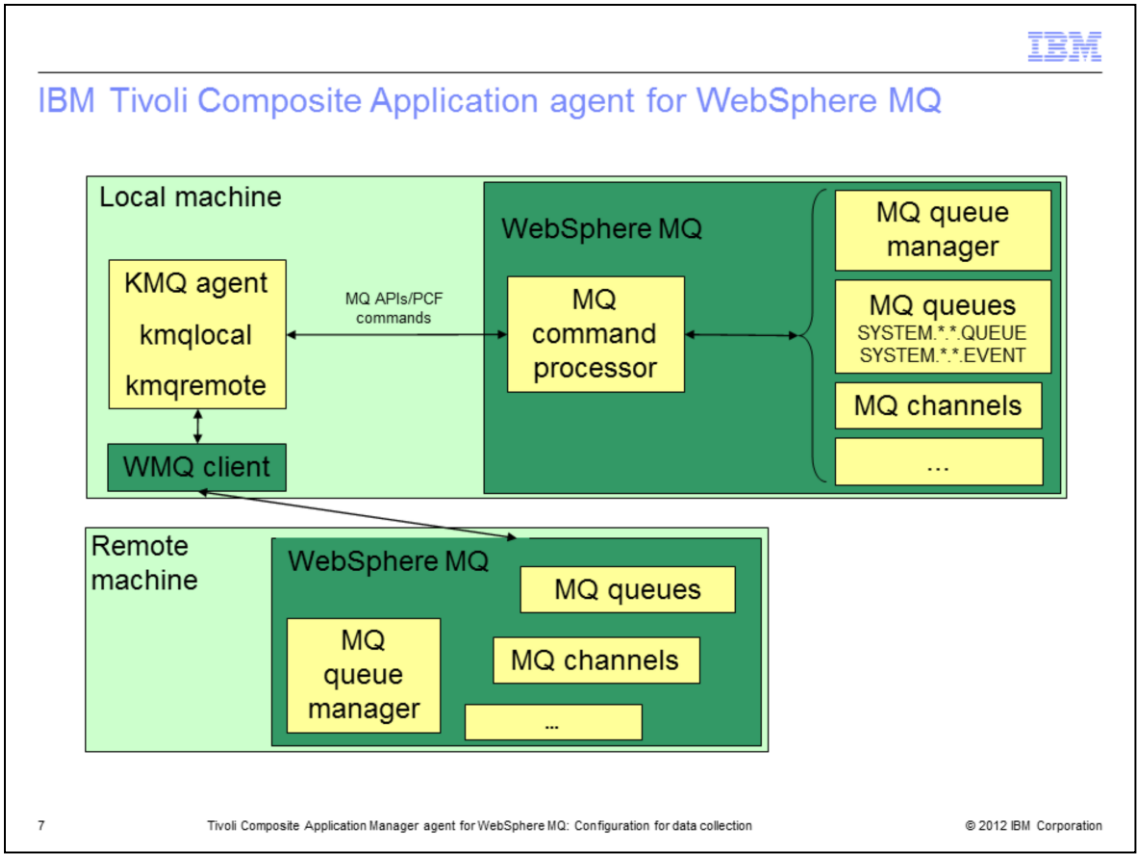


The product code of the MQ agent is KMQ and its FMID in the z/OS® world is HKMQxxx, where xxx is the actual version of the agent, and can be 701, 700, or the old 601.

The MQ agent uses MQ APIs to run PCF commands to gather data and statistics from WebSphere MQ. These commands are processed by the command server and run against the WebSphere MQ itself.

If this process is running, the MQ agent can receive information through the command server, otherwise no information is available.

Notice, however, that the agent does not try to monitor the internal processes of MQ at all.



The image shows the interactions between the MQ agent and WebSphere MQ.

The only interface between the monitoring agent and MQ is the API for delivering commands to the MQ command servers.

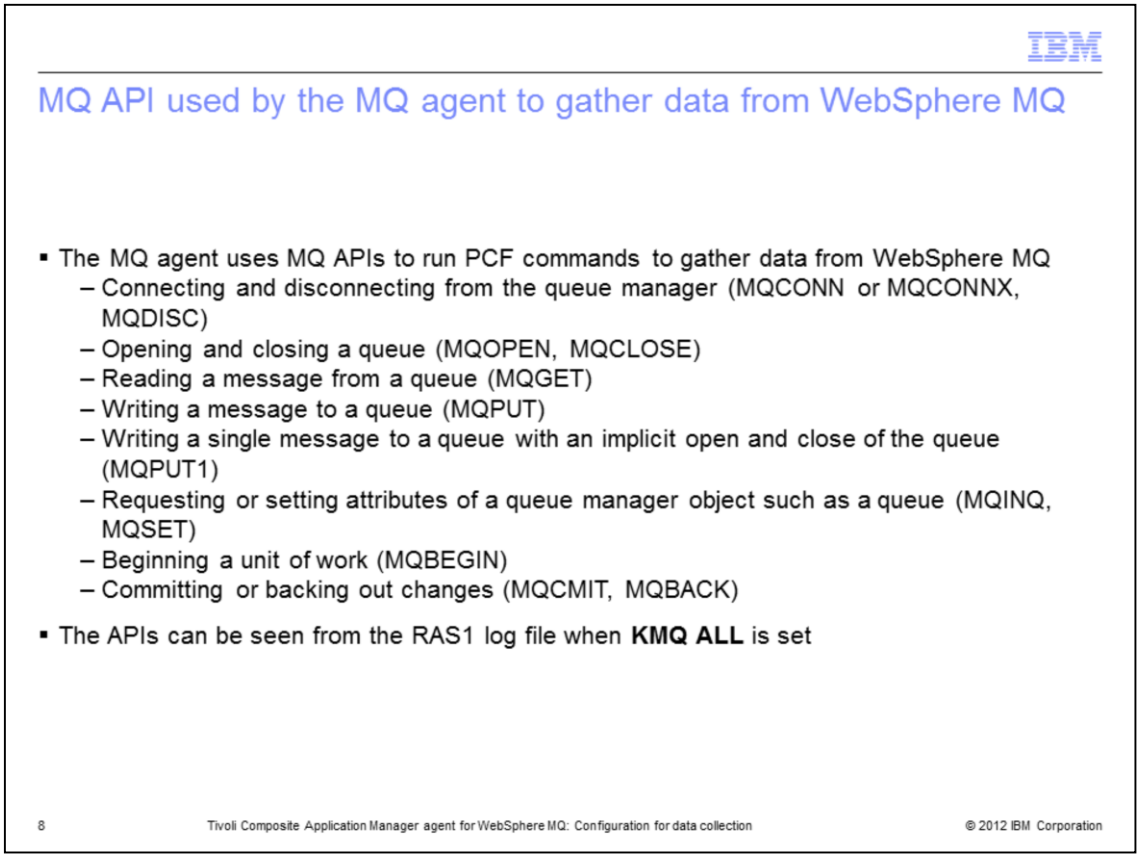
Even for the **amqpcsea** process, which can be considered as the backend of the command server, the MQ agent does not communicate with it or monitor it directly.

The whole communication is through the MQ API.

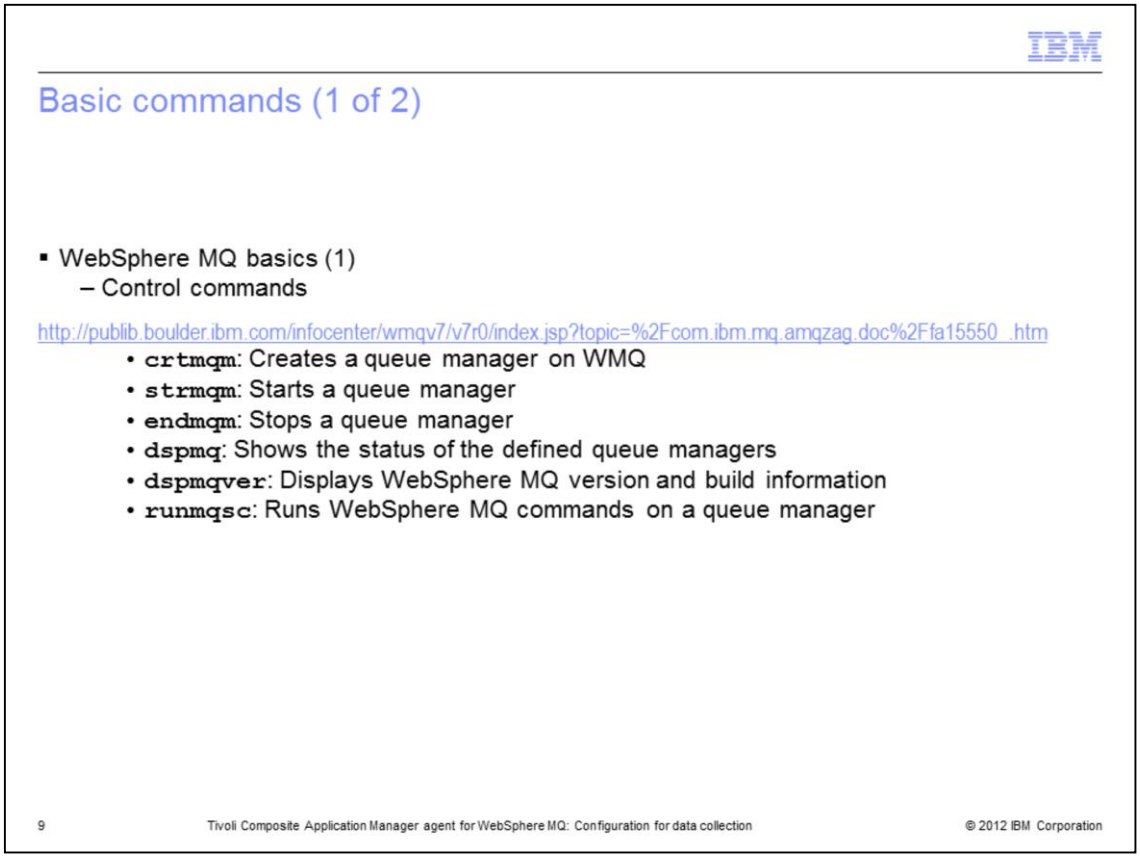
If the **amqpcsea** process is running well, then these API return data and statistics, otherwise they return with a failing code.

Neither direct status inquiry on process nor inter-process communication are included. This non-inclusion reduces the coupling between the MQ products and OMXE MQ itself.

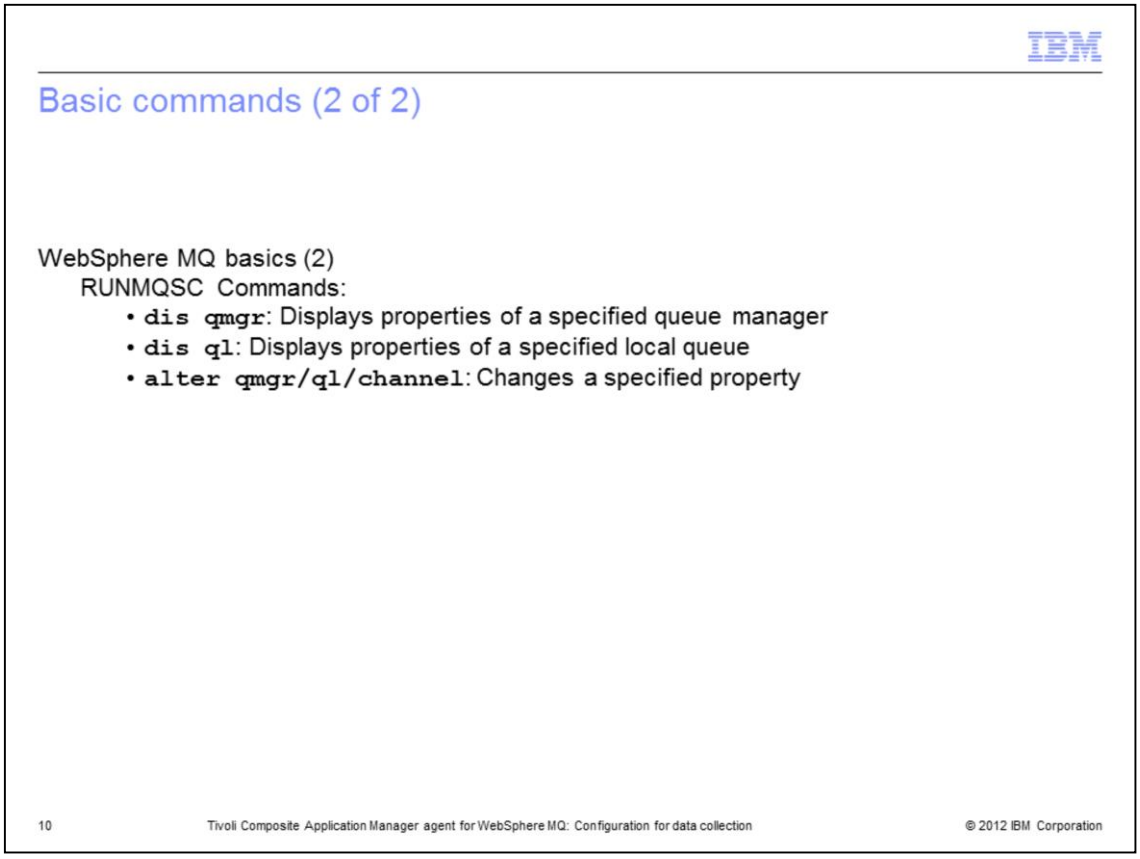
The status changes on the MQ internal processes affect the results returned from the command server. For example, if you can see a channel is running on the Tivoli Enterprise Portal side, a **runmqchl** process must be associated with this channel inside WebSphere MQ. However, the agent receives the information that this channel is running only by communication with the command server with the MQ PCF command API, rather than detecting or direct communication with the **runmqchl** process.



This slide summarizes what happens when the MQ API used by the MQ agent gathers data from WebSphere MQ. The MQ API called for a specific operation from the MQ agent side can be seen from the MQ agent RAS1 log file after having set the **KMQ ALL** trace level.

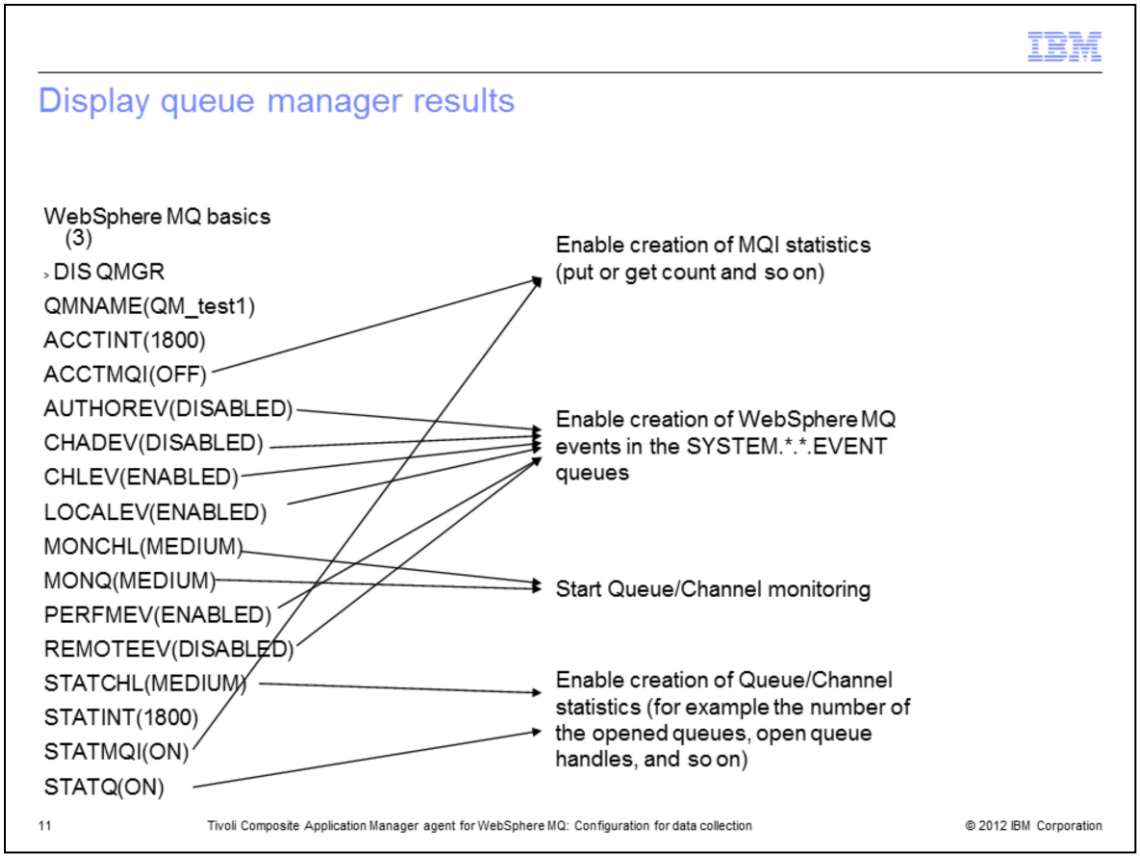


Here are some basic commands for WebSphere MQ itself; they are helpful when working with the MQ agent. Useful control commands are listed, which can be used to create, start and stop queue managers that are then monitored by the MQ agent. An important control command is **runmqsc**, which executes commands against a queue manager.



The MQ agent uses APIs to gather data from a queue manager. The same information can be collected by using the **runmqsc** control commands.

There are many **runmqsc** commands, but the most important are listed here. Using them, you can display queue manager properties and the properties of a queue (in this case of a Local Queue identified by QL). You can also alter the required properties of queue managers, channels, queues, and so on.

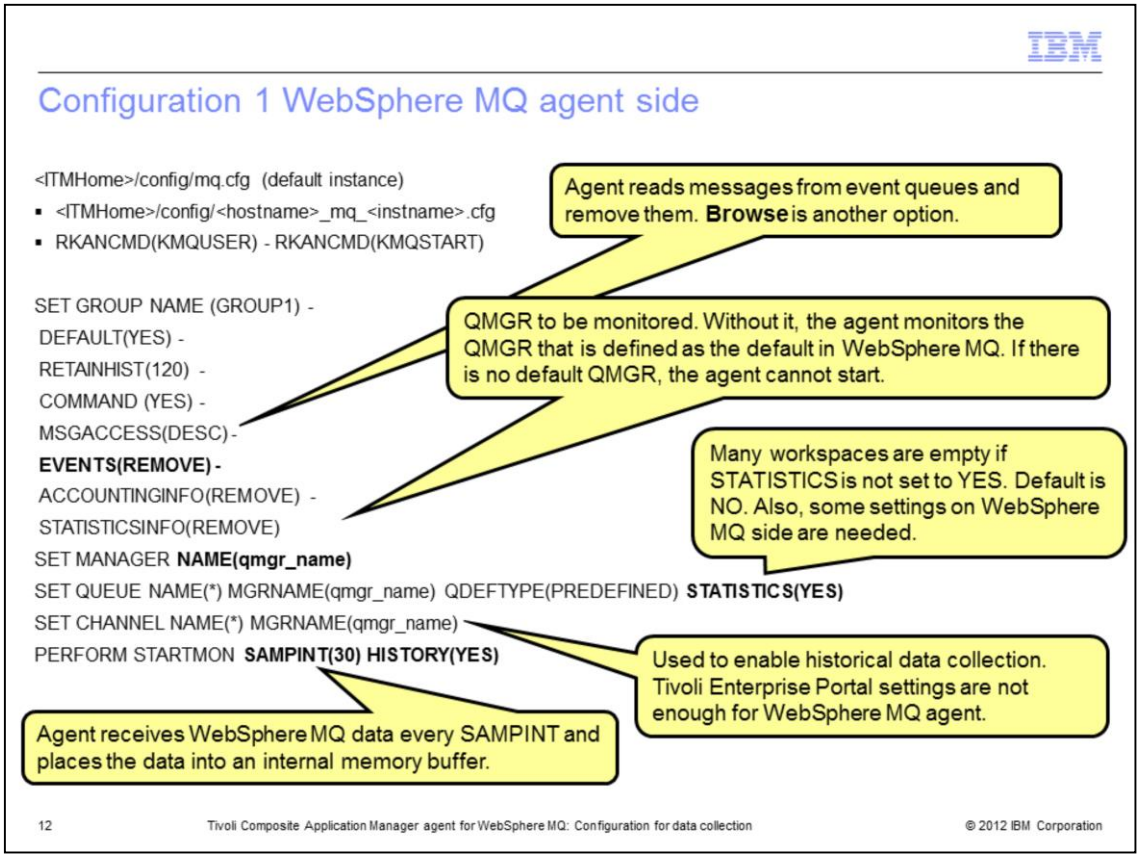


This slide shows the display queue manager results, showing the properties of that queue manager. You can see the meaning of some important parameters.

The STATMQI parameter turns on or off the generation of MQI statistics.

The STATQ or STATCHL parameters enable the generation of queue or channel statistics.

The MONQ or MONCHL parameters start the queue and channel monitoring.



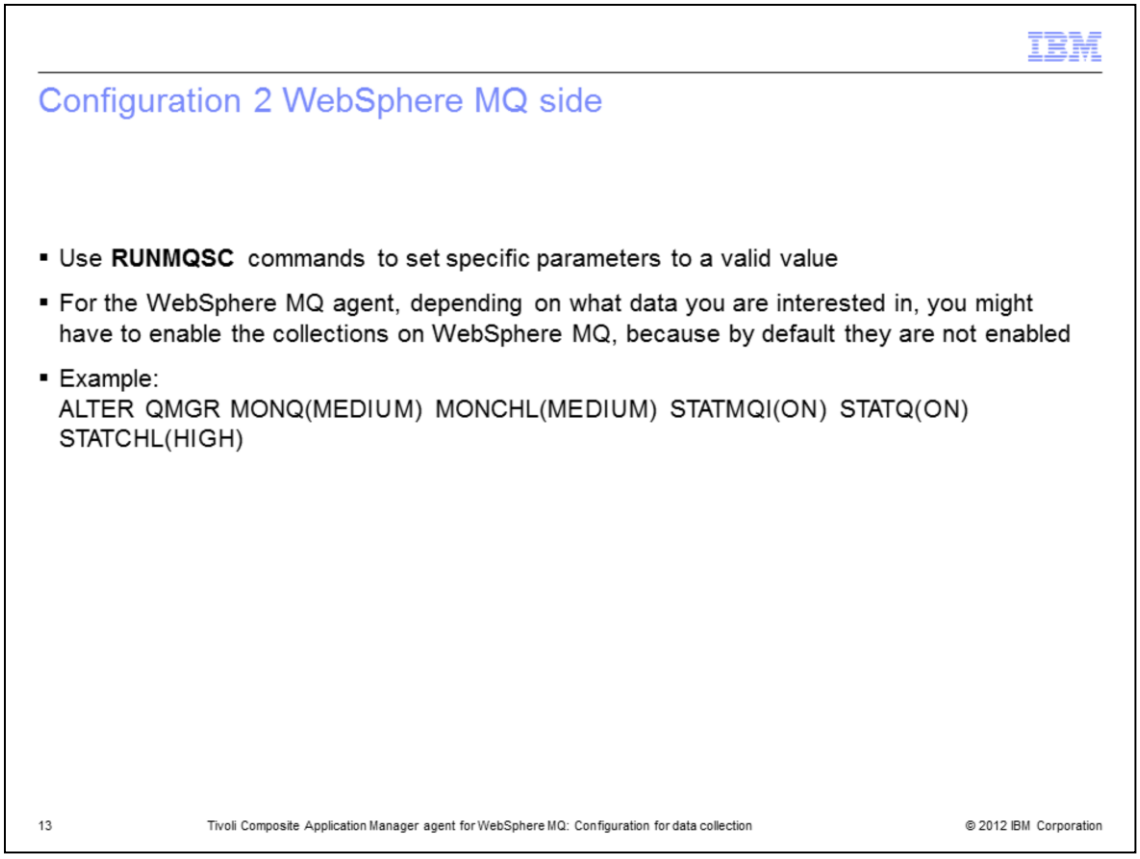
This slide shows the content of a configuration file for the MQ agent with some important parameters explained.

The SET MANAGER NAME(qmgr_name) specifies the queue manager being monitored. Without it, the agent monitors the queue manager defined as **Default** in WebSphere MQ. If there is no default queue manager, the MQ agent cannot start.

The STATISTICS(YES) parameter is important as many workspaces are empty if STATISTICS is not set to YES.

The SAMPINT is the parameter used to indicate to the agent how often it receives WebSphere MQ data and then it places it into the internal memory buffer.

You can see the *MQ agent User's Guide* for a complete description of all the possible parameters.

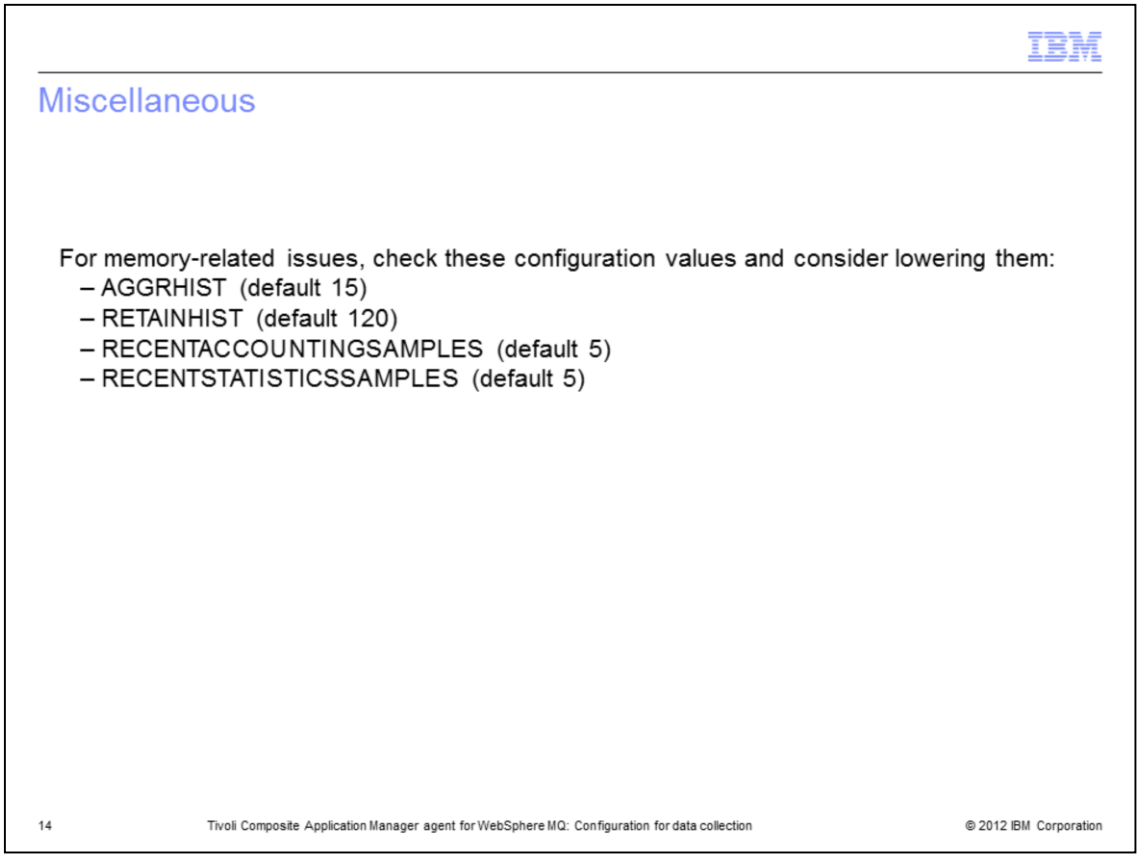


To receive specific information on the Tivoli Enterprise Portal and fill in all of the MQ agent workspaces, some activity is required on the WebSphere MQ side, and some configuration is required.

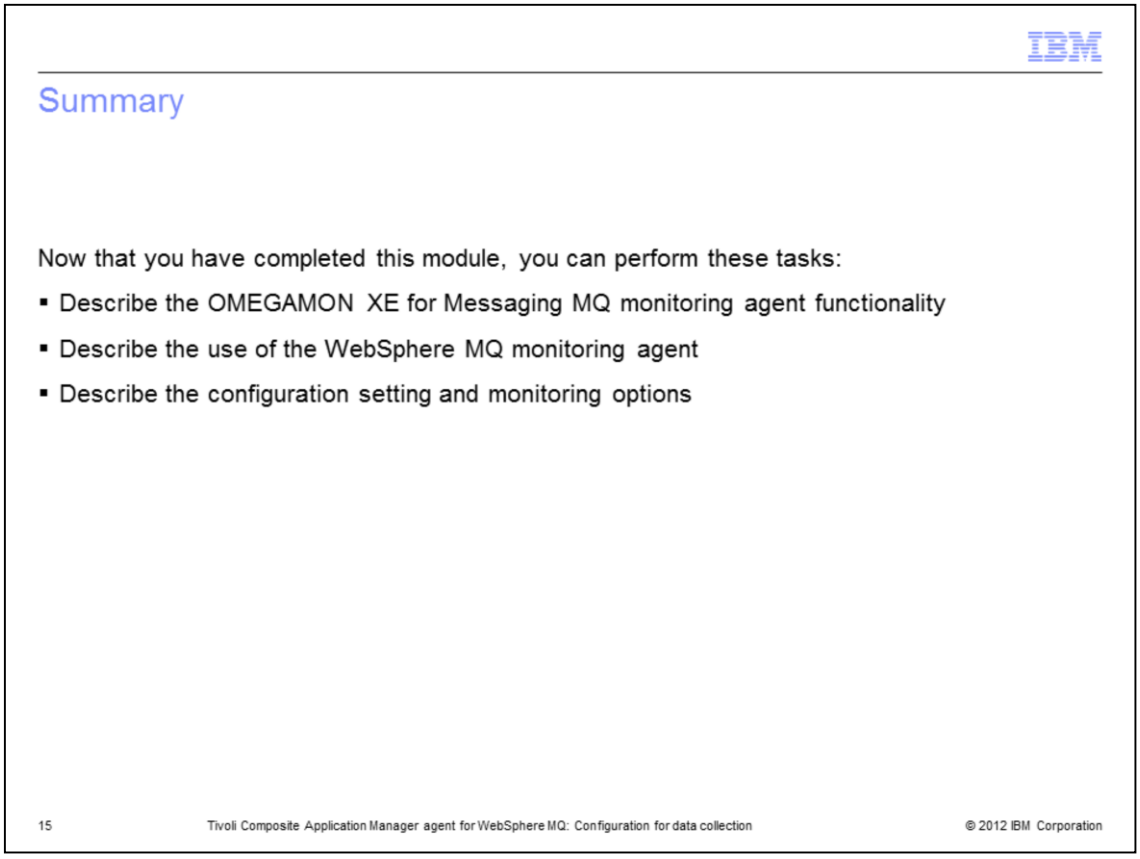
To set the needed parameters, use **RUNMQSC** commands.

Depending on what data are you interested in, you must enable the collections on WebSphere MQ. By default, they are not enabled.

With the command described in the slide, you can enable all of the required statistics on the WebSphere MQ side so that the MQ agent can then gather them using the MQ APIs and report the data on the Tivoli Enterprise Portal.



You can use this reference information in case you have problems with high MQ agent memory consumption.



Summary.

Now that you have completed this module, you can perform these tasks:

- Describe the OMEGAMON XE for Messaging MQ monitoring agent functionality
- Describe the use of the MQ monitoring agent
- Describe the configuration settings and monitoring options

Trademarks, disclaimer, and copyright information