Editors:
W Horn, B Rochwerger

OGSA Logger System
Version 9
(draft)

Status of This Memo

The purpose of this memo is to document some thoughts on issues surrounding the design of an OGSA Logger System. This is a DRAFT document. The editors' intent is to use this memo as a strawman for discussions occurring in the GGF OGSA-WG logging focus area. Distribution of this document is unlimited.

Copyright Notice

Abstract

This document describes an OGSA Logger System, a general purpose component for logging data in a distributed, heterogeneous computing. The system is composed of a set of services that utilize the patterns and semantics defined in the Open Grid Services Infrastructure (OGSI) environment and is intended to support a wide variety of usage scenarios. The interfaces in this document are described using an extended version of the Web Services Description Language (WSDL). The Logger System is designed to virtualize existing logging systems including the zOS System Logger, Windows event logging, and the UNIX syslog facility.

Editors:
W Horn, B Rochwerger

Full Copyright Notice

TBD by GGF

Intellectual Property Statement

TBD by GGF

Contents

1 Introduction

The Open Grid Services Architecture (OGSA) provides a service-oriented approach for dealing with heterogeneous resources in a distributed, loosely-coupled environment [ref:OGSA03]. OGSA services are expressed as Grid services. A Grid service, as defined in the Open Grid Services Infrastructure (OGSI) Specification [ref:OGSI03] supports a minimal set of required interfaces and follows a set of well-defined behaviors. OGSI and OGSA build on the Web services architecture [ref: webservices] by adding concepts taken from Grid computing (e.g., lifecycle management, discovery, security) [ref:gridbook]. Grid service interfaces are expressed using an extension (GWSDL) of the Web Services Description Language (WSDL) [ref:wsdl11].

The importance of a standards-based logging service in a distributed, heterogeneous computing environment is established in [ref:OGSA03]. The current document proposes a set of behaviors and interfaces for an OGSA Logger System. Basically, the system provides a set of services that enables applications to read, write, and manage log records. Like all traditional logging systems it permits the merging of records from multiple instances of multiple applications into a single log stream.

2 Overview

The OGSA Logger System serves as an intermediary between log artifact suppliers and log artifact consumers. Log artifact suppliers store log artifacts that may or may not be read at a later time by log artifact consumers. As documented in [ref:OGSI03], many Grid usage scenarios, including those based on problem determination, metering resource consumption, transaction processing, and security, require logging systems. The logging system requirements generated by these usage scenarios are summarized in the following section.

2.1 Requirements

Applications and processes that consume log artifacts place a number of requirements on OGSA Logger System. These requirements are described in the following subsections.

2.1.1 Legacy Logger Systems

To ensure the general acceptance of the basic log semantics and to enable the exploitation of existing implementations, the OGSA Logger System should support key features found in existing logging implementations (e.g., zOS System Logger, Windows event logging, and UNIX syslog facility).

2.1.2 Persistency

One of the basic goals of any logging system is to provide a mechanism to persist log records. The retention period for log record is determined by consumer requirements. For example, in a real-time monitoring application, data may become irrelevant very fast. Data for an auditing program may be needed months or even years after it was generated. Some applications are best served by circular logs. Other applications depend on persistent log records to recover their state in case of failure. The OGSA Logger System should support a variety of persistency QoSs (e.g., circular logs, logs with policy-specified record retention periods).

2.1.3 Support for Sequential Reads and Writes

The OGSA Logger System should support sequential reads and sequential writes at the granularity of one record representing one log artifact. Support for random access is not required. Byte-level access is not required.

2.1.4 Ordering of Log Records

Logs are frequently used to record a time-ordered sequence of events. The OGSA Logger System should permit consumers to read (time-stamped) records in the same order that they were originally written.

2.1.5 Standard Schema for Log Records

A standard, structured log artifact facilitates the effective intercommunication amongst disparate applications. In some cases, such as when performance is paramount and interoperability is not a concern, less structured artifacts are more appropriate. The OGSA Standard Base Event [ref:CBE03] is designed to accommodate varying degrees of structure and should be used as the basis for the structured log artifact.

2.1.6 Stateful Read Cursor

The OGSA Logger System should permit consumers to sequentially access the records in a log stream using a stateful cursor that is not invalidated by supplier writes or reads from other consumers.

2.1.7 Decoupling of Suppliers from Consumers

The ultimate usage of a log artifact (e.g., audit, system management, problem determination) should be determined, potentially at runtime, by the log artifact consumer not the log artifact supplier. The OGSA Logger System should enable the decoupling of the log artifact supplier from the log artifact consumer.

2.1.8 Broker

The OGSA Logger System should support mechanisms that permit: (1) multiple suppliers to write records into log stream and (2) multiple consumers to read records from a log stream. Furthermore, the system should be able to coordinate concurrent reads and writes. This is a fundamental behavior supported by all production OS logger systems.

2.1.9 Filtering

Frequently the quantity of logging data generated is much greater then the quantity of data actually consumed. To reduce network traffic, the OGSA Logger System should provide a mechanism to filter records as close to the record suppliers as possible.

2.1.10 Merged Log

Some use cases (e.g., transactional logs in parallel systems) require that the logger system exploit underlying merge mechanisms. The OGSA Logger System should expose the capabilities of these merge mechanisms.

2.1.11 Synchronous and Asynchronous Write Semantics

Some applications require an acknowledgement for every write. Other, typically time-sensitive applications, obtain acknowledgements using a callback mechanism. Still other applications do not require any acknowledgement. The OGSA Logger System should support all three patterns.

2.1.12 Duplication of Log Streams

To facilitate the processing of the logs, the OGSA Logger System should provide the capability to create a new log stream by copying records from an existing log stream.

2.1.13 Deletion of Log Records

Log maintenance (e.g., cleanup) requires a mechanism for performing time stamp-based deletes on existing log records in a log stream.

2.1.14 Coexistence with OGSA Messaging

Patterns for the consumption of events vary according to the needs of the consuming applications. In a distributed environment, RPC-based read access to a log stream may not be appropriate for consuming events. For example, a real time monitoring application may need to be asynchronously notified whenever a high-severity artifact is logged (push mode). Another example would be a consumer that polls for metering data every 24 hours (pull mode). These scenarios are better served by using OGSA Messaging System brokers than by using an OGSA Logger System. However, both of the above examples might require that a log be kept of all

events that flow through a messaging broker. The OGSA Logger System should be designed such that a OGSA Messaging System broker can act as logging broker by implementing the Logger interfaces.

2.1.15 Standard Set of Specialized Logstreams

Different applications have different requirements on the features and qualities of service associated with a logstream. To accommodate these demands the OGSA Logger System should support a set of specialized logstreams that somehow span the space of requirements. Specializations should include the following types.

2.1.15.1 Secure Logs

Some applications (e.g., metering, AAA) require that logs be secure (e.g., encrypted).

2.1.15.2 Globally-ordered logs

Some parallel, transaction-oriented applications require merged logs that maintain a global order based on a common clock.

2.1.15.3 Circular logs

For some applications it is preferable to specify retention using a space constraint instead of a time constraint.

2.1.15.4 Duplexed logs

A Log may offer a high reliability QoS by duplexing all writes.

2.1.15.5 Compressed logs

Compressed logs trade performance for space.

2.1.15.6 Compressed logs

Compressed logs trade performance for space.

2.2 Overview of Services

The OGSA Logger System is composed of five services. They are:

- LogManager service
- Log service
- LogStream service
- LogStreamConnection service
- LogBrowseSession service

The portTypes supporting these services and their relationships are shown using UML Figure 1.

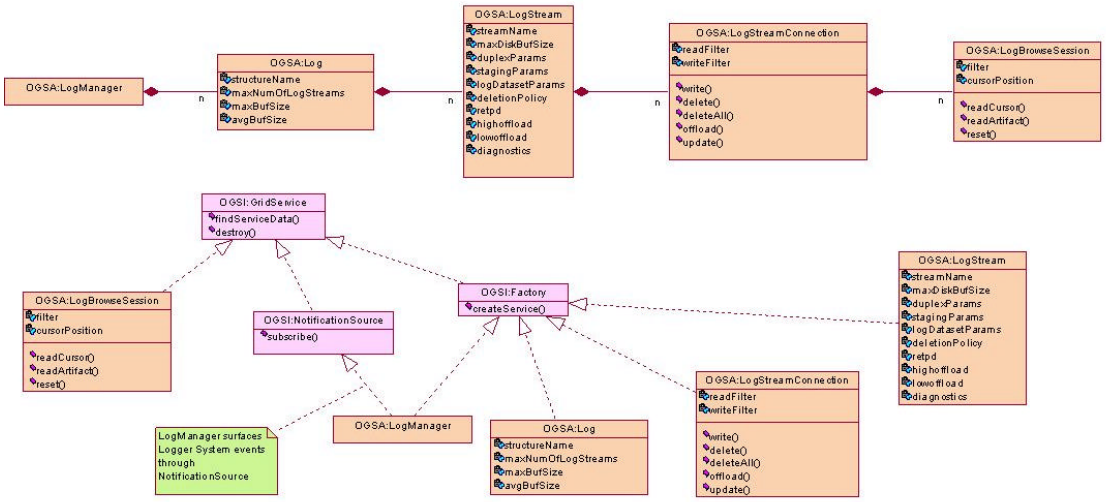

Figure 1: OGSA Logger System Overview

LogManager service

The LogManager service is responsible for managing an instance of the OGSA Logger System. The LogManager service implements the OGSA LogManager portType. The OGSA LogManager portType inherits from the OGSI NotificationSource and OGSI Factory portTypes. The LogManager service acts as a factory for Log service instances. As a NotificationSource, it is responsible for surfacing all OGSA Logger System state changes, providing a single point of contact for obtaining all management-related events.

Log service

The Log service implements the OGSA Log portType. The OGSA Log portType inherits from the OGSI Factory portType. The number of LogStream instances that can be created is specified when a Log service is instantiated (using the factory operations in LogManager portType). If the underlying system supports a hardware-assisted merge facility then the details of the connection to the enabling hardware (e.g., zSeries Coupling Facility) are specified when the Log service is instantiated. The Log service acts as a factory for LogStream service instances.

LogStream service

The LogStream service implements the OGSA LogStream portType. The OGSA LogStream portType inherits from the OGSI Factory portType. A relatively long list of parameters is available to configure a particular LogStream service instance. LogStream policies for retention, offload, persistence, availability, and staging are defined by these parameters. The LogStream service acts as a factory for LogStreamConnection service instances.

LogStreamConnection service

All access (read or write) to the records in a LogStream are made through the LogStreamConnection service. The LogStreamConnection service implements the OGSA LogStreamConnection portType. The OGSA LogStreamConnection portType inherits from the OGSI Factory portType. The LogStreamConnection portType has operations which permit the writing and importing of log records. Additionally, the LogStreamConnection service acts as a factory for LogBrowseSession service instances.

LogBrowseSession service

An instance of the LogBrowseSession service is used to read log records from a LogStream. The LogBrowseSession service implements the OGSA LogBrowseSession portType. The OGSA LogBrowseSession portType inherits from the OGSI GridService portType. Each LogBrowseSession maintains a cursor for navigating its corresponding LogStream. The LogBrowseSession portType has operations to access records by id or by timestamp, position its cursor, or to use its cursor to read a sequence of records.

2.3 Relationship to other OGSA Specifications

TBD.

2.3.1 OGSA Messaging Service

TBD.

2.3.2 OGSA Standard Base Event

TBD.

2.3.3 OGSA Policy Service

TBD.

2.3.4 OGSA Security Service

TBD.

3 LogManager Service

Each instance of a LogManager service is responsible for managing an instance of the OGSA Logger System. The LogManager acts as a factory for Log service instances and is responsible for surfacing all OGSA Logger System state changes.

The LogManager portType extends the OGSI Factory portType and the OGSI NotificationSource portType both of which in turn extend the OGSI GridService portType. The NotificationSource portType is used to provide a single point of contact for obtaining all management events emitted by the Logger System. The LogManager portType introduces no additional operations.

3.1 LogManager: GridService

3.1.1 LogManager: GridService:: findServiceData

The Log portType adds one service data element. TBD, here are some notes:

- logsnum

This attribute specifies the number of Log instances that can be allocated. If the value of this attribute is 0 then the upper limit on the number of Log instances that can be allocated is not specified. On input, this attribute is optional and has a default value of 0.

3.1.2 LogManager: GridService:: setServiceData

Not supported.

3.1.3 LogManager: GridService:: requestTerminationAfter

Persistent service, can't be terminated.

3.1.4 LogManager: GridService:: requestTerminationBefore

Persistent service, can't be terminated.

3.1.5 LogManager: GridService:: destroy

Persistent service, can't be terminated.

3.2 LogStream: Factory

The LogManager service is a factory for the creation of Log instances.

3.2.1 LogManager: Factory:: createService

createService TBD.

3.3 LogManager: NotificationSource

The LogManager service uses the NotificationSource portType to provide a single point of contact for obtaining all management events emitted by the OGSA Logger System.

3.3.1 LogManager: NotificationSource:: subscribe

TBD.

3.4 LogManager Operations

None.

4 Log Service

Instances of a Log service are created by the LogManager service. If the system underlying the Logger System supports a hardware-assisted merge facility then the details of the connection to the enabling hardware are specified when the Log service is instantiated. The Log service acts as a factory for LogStream service instances.

The Log Service implements the Log portType. The Log portType extends the OGSI Factory portType which in turn extends the OGSI GridService portType. The Log portType introduces no additional operations.

4.1 Log: GridService

4.1.1 Log: GridService:: findServiceData

As shown in Figure 2, the Log portType adds one new service data element, <log:Log>.

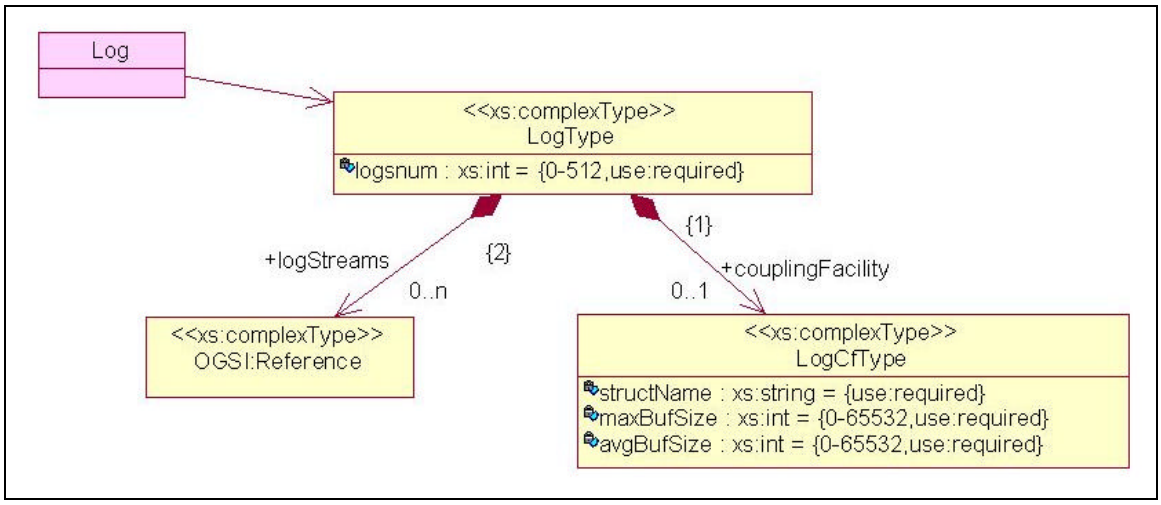

Figure 2: Log service data element

The service data element <Log> is of type <LogType>.

<LogType>

The following attributes are defined in the <LogType> element type:

- logsnum

  This attribute specifies the number of LogStream instances that can be allocated. If the value of this attribute is 0 then the upper limit on the number of LogStream instances is not provided.

The <LogType> element type contains [0…1] <couplingFacility> elements and [0…n] <logStreams> elements. The <couplingFacility> element is of type <LogCfType>, if it is present then the Log is connected to an underlying coupling facility. The <logStreams> element is of type <ogsi:reference> and contains the OGSI locators for LogStream service instances created by the Log service. The <ogsi:reference> element type is defined in [ref:ogsi].

<LogCfType>

The following attributes are defined in the <LogCfType> element type:

- structName

This attribute specifies the name of the underlying coupling facility structure.

- maxBufSize

  This attribute specifies the largest log block size that the coupling facility can handle. If the value of the attribute is 0 then there is no upper limit on the block size.

- avgBufSize

  This attribute specifies the average buffer size and is used to optimize the connection to the underlying coupling facility. If the value of the attribute is 0 then the coupling facility either does not use this attribute for optimization or chooses not to expose it.

4.1.2 Log: GridService:: setServiceData

Not supported.

4.1.3 Log: GridService:: requestTerminationAfter

Supported. Details TBD.

4.1.4 Log: GridService:: requestTerminationBefore

Supported. Details TBD.

4.1.5 Log: GridService:: destroy

Supported. Details TBD.

4.2 Log: Factory

The Log service is a factory for the creation of LogStream instances.

4.2.1 Log: Factory:: createService

createService TBD.

4.3 Log Operations

None.

5 LogStream Service

Instances of a LogStream service are created by the Log service. A LogStream is controlled by policies for retention, offload, persistence, availability, and staging. The LogStream service acts as a factory for LogStreamConnection service instances.

The LogStream service implements the LogStream portType. The LogStream extends the OGSI Factory portType which in turn extends the OGSI GridService portType.

5.1 LogStream: GridService

5.1.1 LogStream: GridService:: findServiceData

As shown in Figure 3, the LogStream portType adds one new service data element, <log:LogStream>.

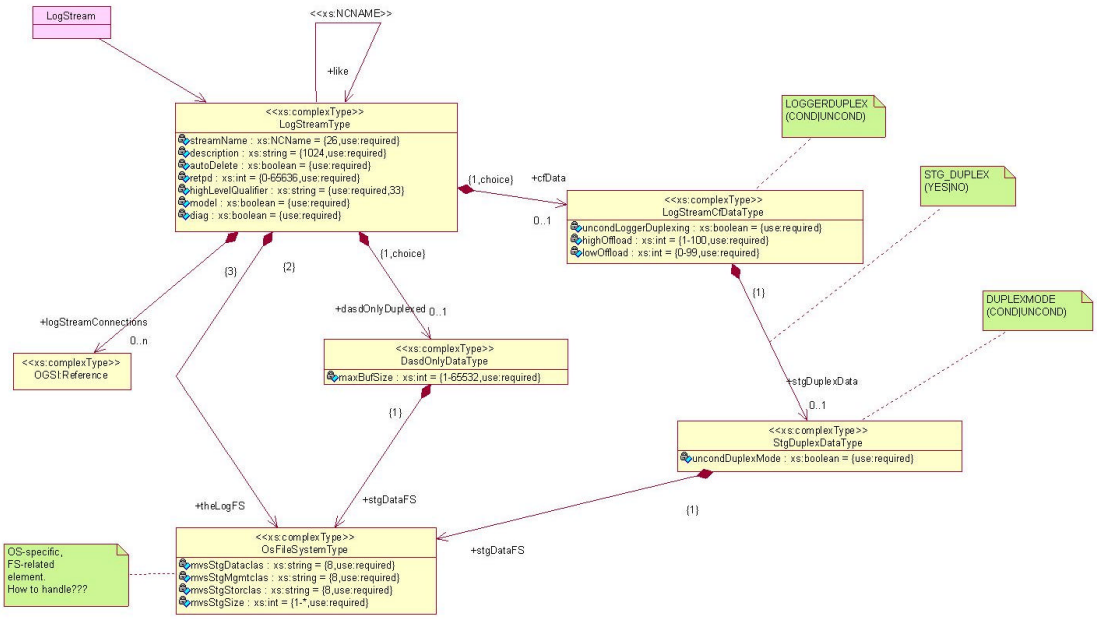

Figure 3: LogStream service data element

The service data element <LogStream> is of type <LogStreamType>.

<LogStreamType>

The following attributes are defined in the <LogStreamType> element type:

- streamName

  Globally unique name for this LogStream.

- description

  This attribute contains a user-defined description of the log stream. On input, this attribute is optional and has a default value of an empty string.

- autoDelete

If the value of this attribute is false then the LogStream will physically delete an entire log data set only when both of the following conditions are true:

- The delete operation of the LogStreamConnection has been used to mark an record for deletion.
- The retention period for all the records in the log data set has been exceeded.

If the value of this attribute is true then the LogStream will physically delete an entire log data set when either of the above conditions are true.

- retpd

  The value of this attribute specifies the retention period, in days, for records.

- highLevelQualifier

  The value of this attribute specifies the high level qualifier for both the log stream data set name and the staging data set name.

- like

  The streamName of the LogStream that this LogStream models.

- model

  If the value of this attribute is true, then this LogStream is a model for other LogStreams.

- diag

  If the value of this attribute is true then dumping or additional diagnostics will be provided by Logger for certain conditions.

The <logStreamType> element type contains either a <cfData> element or a <dasdOnlyDuplexed> element, a <theLogFS> element, and [0…n] <logStreamConnections> elements. The <cfData> element is of type <LogStreamCfDataType>, if it is present then the Log is connected to an underlying coupling facility. The <dasdOnlyDuplexed> element is of type <DasdOnlyDataType>, if it is present then the LogStream is not connected to a coupling facility. The <logStreamConnections> element is of type <ogsi:reference> and contains the OGSI locators for LogStreamConnection service instances created by the LogStream. The <theLogFS> element is of type <OsFileSystemType> and contains the OS-specific location of the log data set. The <ogsi:reference> element type is defined in [ref:ogsi].

<LogStreamCfDataType>

The following attribute is defined in the <LogStreamCfDataType> element type:

- uncondLoggerDuplexing

  If this attribute has a value of true then the LogStream will unconditionally provide its own duplexing of the log data regardless of any other duplexing capabilities (such as structure system-managed duplexing rebuild) that may be present. If this attribute has a value of false then the LogStream will duplex log data only if there is no alternative duplexing configuration that provides an equivalent or better recoverability of the log data.

- highOffload

  The value of this attribute specifies the percent value to be used as a high offload threshold for the underlying coupling facility associated with LogStream. This value is always greater than lowOffload. The default value is 80%.

- lowOffload

The value of this attribute specifies the percent value to be used as a low offload threshold for the underlying coupling facility associated with this LogStream. This value must be less than highOffload. The default value is 0%.

The <logStreamCfDataType> element type contains [0…1] <stgDuplexData> elements. The <stgDuplexData> element is of type <StgDuplexDataType>, if it is present then the LogStream data for a coupling facility is duplexed in the DASD staging data sets.

<StgDuplexDataType>

The following attribute is defined in the <StgDuplexDataType> element type:

- uncondDuplexMode

  If this attribute has a value of true the LogStream always duplexes log data to its staging data sets. If this attribute has a value of false then log data is duplexed into DASD staging data sets only if the connectivity to the coupling facility contains a single point of failure.

The <StgDuplexDataType> element type contains 1 <stgDataFS> element. The <stgDataFS> element is of type <OsFileSystemType> and contains filesystem specifics for the staging dataset.

<OsFileSystemType>

The following attributes are defined in the <OsFileSystemType> element type:

- mvsStgDataclas (MVS Only)

  This attribute specifies the name of SMS data class that will be used for allocation of the DASD staging data set for this log stream.

- mvsStgMgmtclas (MVS Only)

  This attribute specifies the name of SMS management class that will be used for allocation of the DASD staging data set for this log stream.

- mvsStgStorclas (MVS Only)

  This attribute specifies the name of SMS storage class that will be used for allocation of the DASD staging data set for this log stream.

- mvsStgSize (MVS Only)

  This attribute specifies the size in 4K blocks of the DASD staging data set for the LogStream.

<DasdOnlyDataType>

The following attribute is defined in the <DasdOnlyDataType> element type:

- maxBufSize

  This attribute represents the largest log block size that the LogStream can handle

The < DasdOnlyDataType > element type contains 1 <stgData> element. The <stgData> element is of type <StgDataType>. The <stgDataFS> element is of type <OsFileSystemType> and contains filesystem specifics for the staging dataset.

5.1.2 LogStream: GridService:: setServiceData

Supported. Details TBD.

5.1.3 LogStream: GridService:: requestTerminationAfter

Supported. Details TBD.

5.1.4 LogStream: GridService:: requestTerminationBefore

Supported. Details TBD.

5.1.5 LogStream: GridService:: destroy

Supported. Details TBD.

5.2 LogStream: Factory

The LogStream service is a factory for the creation of LogStreamConnection instances.

5.2.1 LogStream: Factory:: createService

createService TBD.

5.3 LogStream Operations

None.

6 LogStreamConnection Service

Instances of a LogStreamConnection service are created by the LogStream service. All access (read or write) to the records in a LogStream are made through the LogStreamConnection service. The LogStreamConnection service acts as a factory for LogBrowseSession service instances.

The LogStreamConnection service implements the LogStreamConnection portType. The LogStreamConnection portType extends the OGSI Factory portType which in turn extends the OGSI GridService portType, it also introduces additional operations.

6.1 LogStreamConnection: GridService

6.1.1 LogStreamConnection: GridService:: findServiceData

As shown in Figure 4, the LogStreamConnection portType adds one new service data element, <log:LogStreamConnection>.

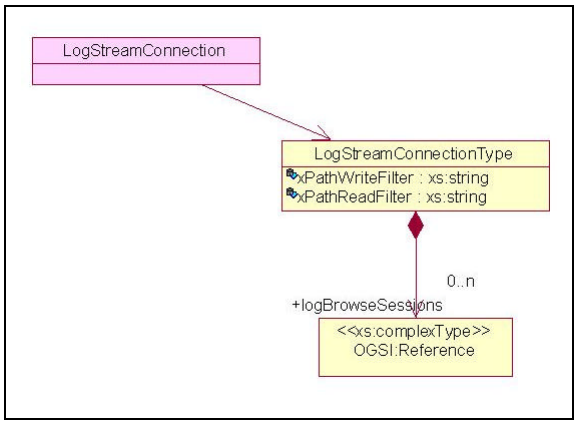

Figure 4: LogStreamConnection service data element

The service data element <LogStreamConnection> is of type <LogStreamConnectionType>.

<LogStreamConnectionType>

The following attributes are defined in the <LogStreamConnectionType> element type:

- xPathWriteFilter

  This attribute specifies the XPath expression used for any write made through this connection.

- xPathReadFilter

  This attribute specifies the XPath expression used for any read made through this connection.

The <LogStreamConnectionType> element type contains [0…n] <logBrowseSession> elements. The <logBrowseSession> element is of type <ogsi:reference> and contains the OGSI locators for LogBrowseSession service instances created by the LogStreamConnection. The <ogsi:reference> element type is defined in [ref:ogsi].

6.1.2 LogStreamConnection: GridService:: setServiceData

Not supported? Maybe for filters?

6.1.3 LogStreamConnection: GridService:: requestTerminationAfter

Supported. Details TBD.

6.1.4 LogStreamConnection: GridService:: requestTerminationBefore

Supported. Details TBD.

6.1.5 LogStreamConnection: GridService:: destroy

Supported. Details TBD.

6.2 LogStreamConnection: Factory

The LogStreamConnection service is a factory for the creation of LogBrowseSession instances.

6.2.1 LogStreamConnection: Factory:: createService

createService TBD.

6.2.2 LogStreamConnection: Operations

6.2.2.1 LogStreamConnection:: write

This operation is used to write a log record to a LogStream. Figure 5 shows the GWSDL for the write operation.

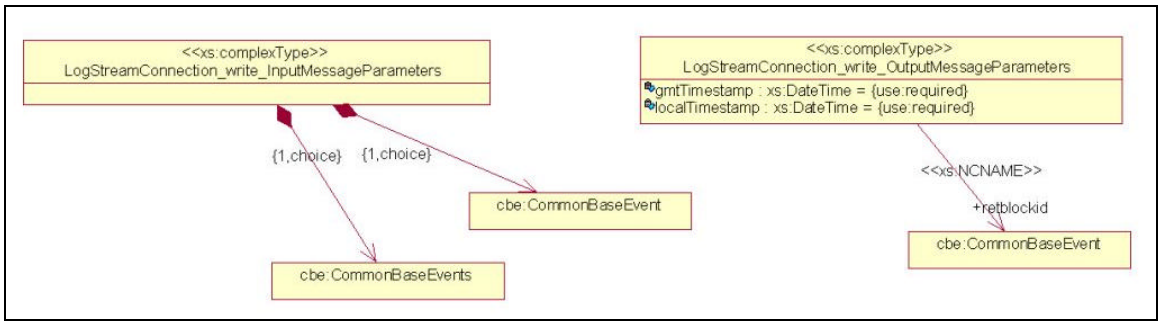

Figure 5: GWSDL for LogStreamConnection::write

TBD, here are some notes:

- buffer

This input attribute specifies the log record.

- retblockid

Assuming a successful write, this output attribute contains a unique identifier for the log record written by this operation.

- gmtTimestamp

Assuming a successful write, this output attribute contains the GMT timestamp for the log record written by this operation.

- localTimestamp

Assuming a successful write, this output attribute contains the local timestamp for the log record written by this operation.

6.2.2.2 LogStreamConnection:: deleteAll

This operation is used to delete all the log records from a LogStream. Figure 6 shows the GWSDL for the deleteAll operation.

Figure 6: GWSDL for LogStreamConnection::deleteAll

6.2.2.3 LogStreamConnection:: delete

This operation is used to delete a range of log records from a LogStream. Figure 7 shows the GWSDL for the delete operation.

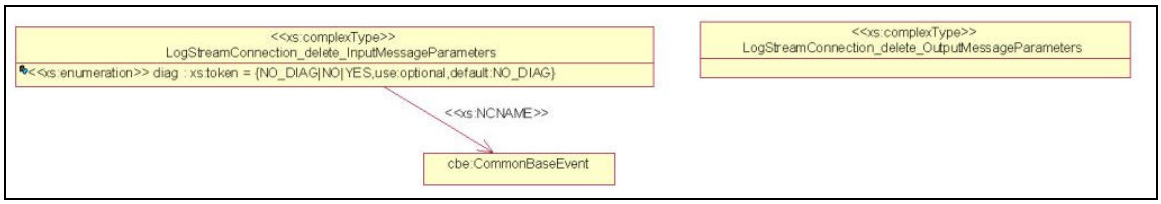

Figure 7: GWSDL for LogStreamConnection::delete

TBD, here are some notes:

- blockid

This input attribute identifies a log record by specifying its unique identifier. All records older than the specified record are deleted.

6.2.2.4 LogStreamConnection:: import

This operation is used to import log records from one log stream to another. For now, use the GWSDL. Details TBD.

6.2.2.5 LogStreamConnection:: update

This operation specifies that any future log blocks written to the log stream cannot have a time stamp less than the updated time stamp. Figure 8 shows the GWSDL for the update operation.

Figure 8: GWSDL for LogStreamConnection::update

6.2.2.6 LogStreamConnection:: offload

This operation forces an offload of the log records from the coupling facility or the duplexed dasd to the log. This operation has no effect if a coupling facility or duplexed dasd is not used by the LogStream. Figure 9 shows the GWSDL for the offload operation.

Figure 9: GWSDL for LogStreamConnection::offload

7 LogBrowseSession Service

Instances of a LogBrowseSession service are created by the LogStreamConnection service. An instance of the LogBrowseSession service is used to read log records from a LogStream. Each LogBrowseSession maintains a cursor for navigating its corresponding LogStream and has operations to access records by id or by timestamp, position its cursor, or to use its cursor to read a sequence of records.

The LogBrowseSession service implements the LogBrowseSession portType. The LogBrowseSession portType extends the OGSI Grid service portType, it also introduces additional operations.

7.1 LogBrowseSession: GridService

7.1.1 LogBrowseSession: GridService:: findServiceData

As shown in Figure 10, the LogBrowseSession portType adds one new service data element, <log: LogBrowseSession >.

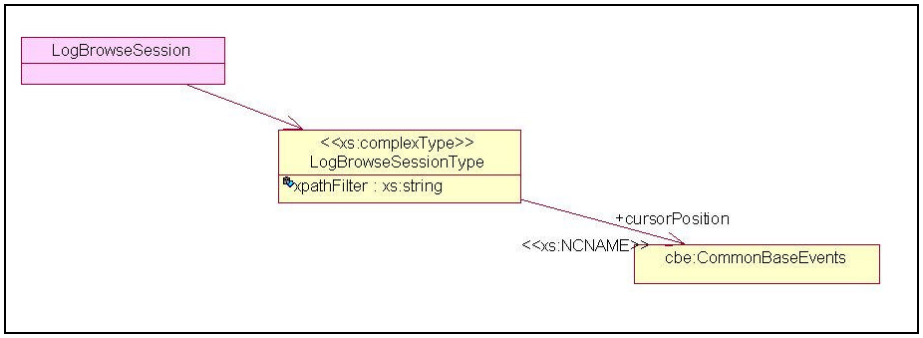

Figure 10: LogBrowseSession service data element

The service data element <LogBrowseSession> is of type <LogBrowseSessionType>.

<LogBrowseSessionType>

The following attributes are defined in the <LogBrowseSessionType> element type:

- xPathFilter

  This attribute specifies the XPath expression used for any read made during this session. In addition to this filter, reads are also potentially filtered by the parent connection.

- cursorPosition

  The value of this attribute contains the unique identifier and indicates the current position of the cursor.

7.1.2 LogBrowseSession: GridService:: setServiceData

Not supported.

7.1.3 LogBrowseSession: GridService:: requestTerminationAfter

Supported. Details TBD.

7.1.4 LogBrowseSession: GridService:: requestTerminationBefore

Supported. Details TBD.

7.1.5 LogBrowseSession: GridService:: destroy

Supported. Details TBD.

7.1.6 LogBrowseSession: Operations

7.1.6.1 LogBrowseSession:: readCursor

This operation is used to read a sequence of one or more records from the log stream. Figure 11 shows the GWSDL for the readCursor operation.

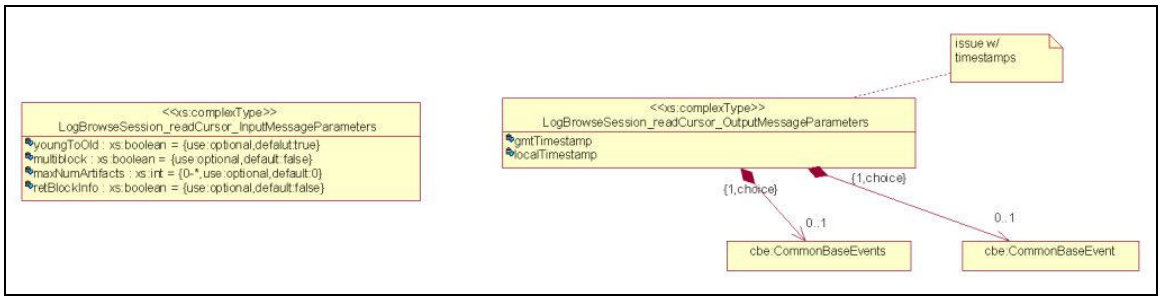

Figure 11: GWSDL for LogBrowseSession::readCursor

TBD, here are some notes:

- youngToOld

If the value of this input attribute is true then the cursor moves from young to old. Otherwise, the cursor moves from old to young. On input, this attribute is optional and has a default value of true.

- multiblock

If the value of this input attribute is true then the service will return up to maxNumRecords records. On input, this attribute is optional and has a default value of false.

- maxNumRecords

This attribute is only relevant if multiblock has a value of true. The value of this input attribute represents the maximum number of records that will be returned for a multiblock request. If the value of this attribute is 0 then there is no upper limit. On input, this attribute is optional and has a default value of 0.

- retBlockInfo

If the value of this input attribute is true then the record xml will contain identification attributes such as the unique identifier, timestamp. Otherwise, identification information is not returned. On input, this attribute is optional and has a default value of false.

- record

Each instance of this output attribute contains a requested log record.

- gmtTimestamp

This output attribute contains the GMT timestamp for the log record.

- localTimestamp

This output attribute contains the local timestamp for the log record.

- retBlockId

This output attribute contains the unique identifier for the log record.

7.1.6.2 LogBrowseSession:: readBlock

This operation is used to read a specific log record from a LogStream. Figure 12 shows the GWSDL for the readBlock operation.

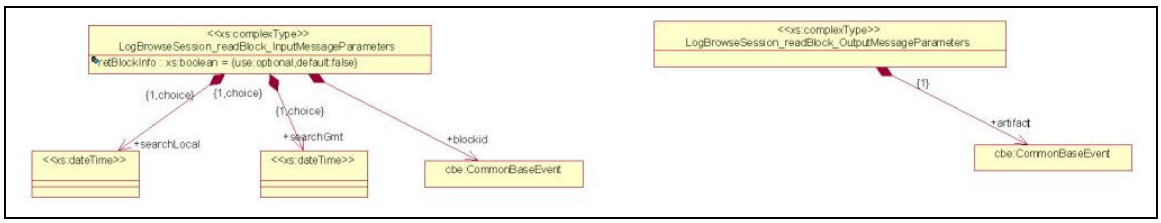

Figure 12: GWSDL for LogBrowseSession::readBlock

TBD, here are some notes:

The following three attributes, blockid, searchGmt, and searchLocal, are mutually exclusive, that is, only one of them can be present.

- blockid

The value of this input attribute represents the unique identifier of the requested log record.

- searchGmt

The value of this input attribute represents the GMT timestamp of the requested log record. The search is performed from youngest to oldest. If no exact match is found, this operation will return the next latest (youngest) record.

- searchLocal

The value of this input attribute represents the local timestamp of the requested log record. The search is performed from youngest to oldest. If no exact match is found, this operation will return the next latest (youngest) record.

- retBlockInfo

If the value of this input attribute is true then the record xml will contain identification attributes such as the unique identifier, timestamp. Otherwise, identification information is not returned. On input, this attribute is optional and has a default value of false.

- record

Each instance of this output attribute contains a requested log record.

- gmtTimestamp

This output attribute contains the GMT timestamp for the log record.

- localTimestamp

This output attribute contains the local timestamp for the log record.

- retBlockId

This output attribute contains the unique identifier for the log record.

7.1.6.3 **LogBrowseSession:: reset**

This operation is used to reset the LogBrowseSession cursor to either the oldest or youngest record in the LogStream. Figure 13 shows the GWSDL for the reset operation.

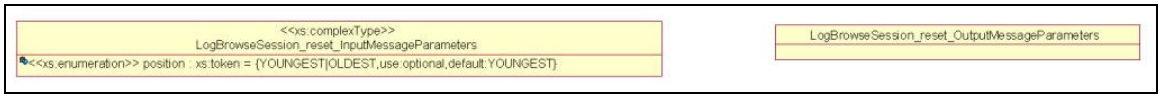

Figure 13: GWSDL for LogBrowseSession::reset

TBD, here are some notes:

- position

The value of this input attribute is either “youngest” or “oldest”. If the value is “oldest” the cursor is positioned at the oldest log record in the log stream. If the value is “youngest” the cursor is positioned at the youngest log record in the log stream. On input, this attribute is optional and has a default value of “youngest”.

8 Logger System Service Interactions

This section explains how record suppliers and consumers interact with the OGSA Logger System. Details TBD.

8.1 Setting up a Log Stream

Details TBD.

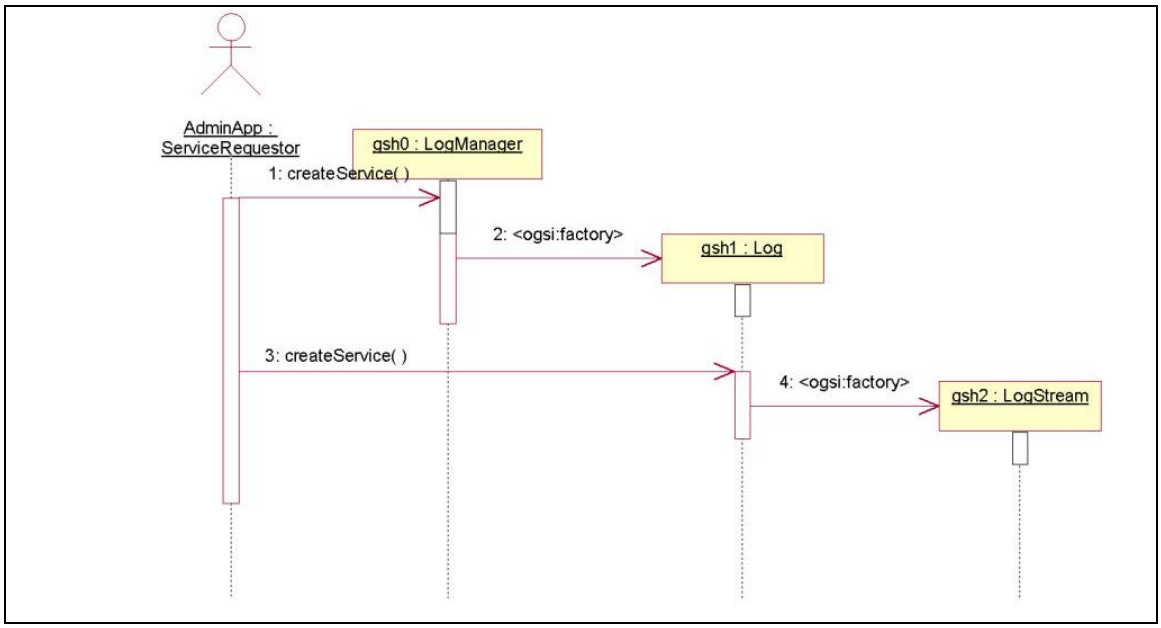

Figure 14: Setting up a Log Stream

8.2 Writing to a Log Stream

Details TBD.

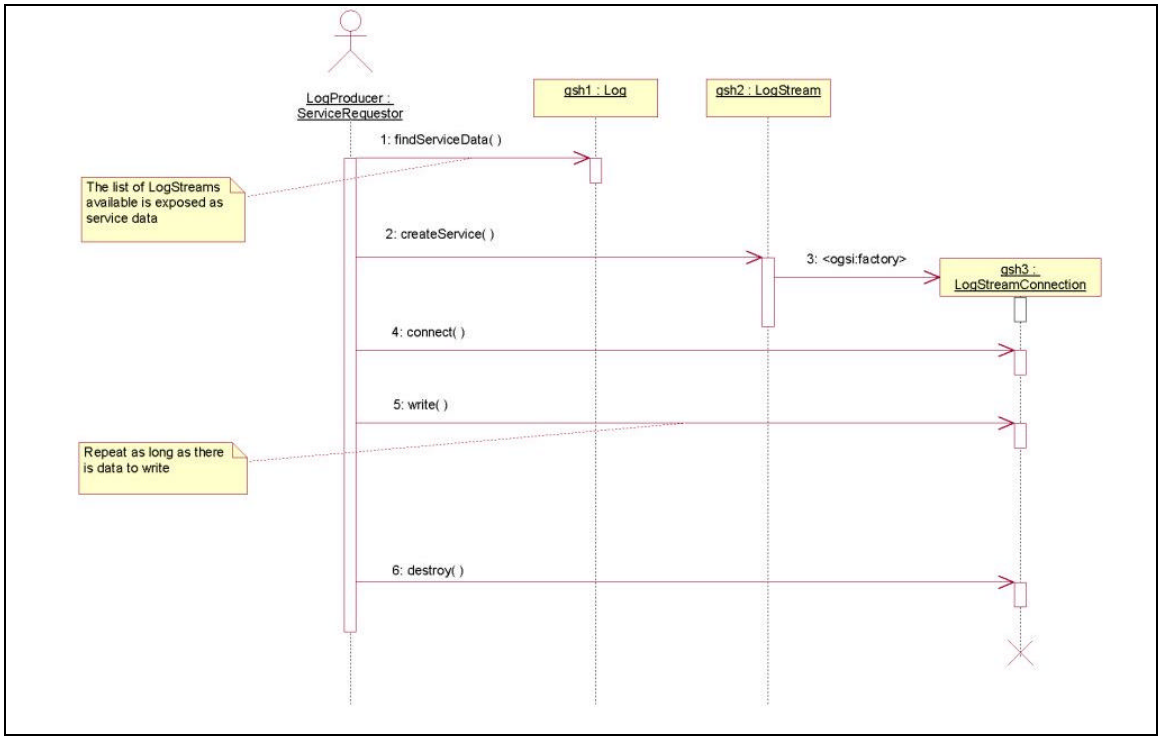

Figure 15: Writing to a Log Stream

8.3 Reading from a Log Stream

Details TBD.

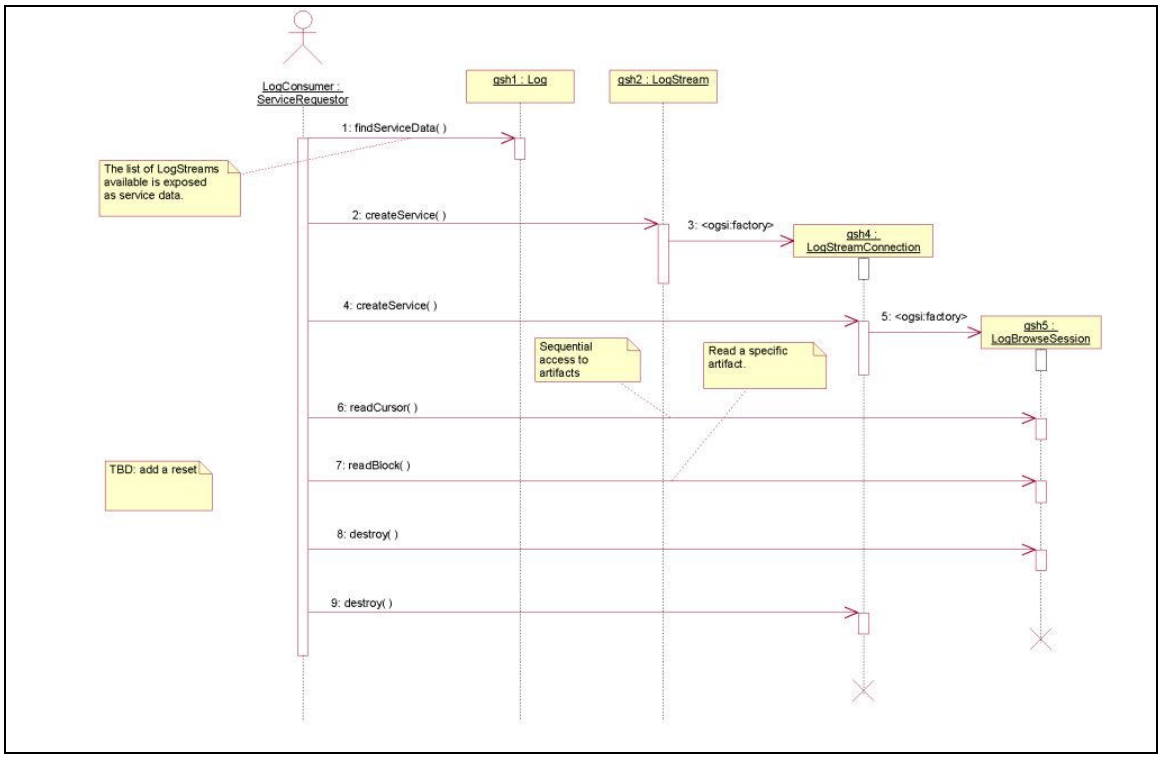

Figure 16: Reading from a Log Stream

9 Security Considerations

This is a REQUIRED section. TBD by Frank Siebenlist.

10 Glossary

Recommended but not required. TBD.

11 Editor Information

- W. Horn hornwp@us.ibm.com
- B. Rochwerger rochwer@il.ibm.com

12 Contributors

The editors would like to acknowledge the contributions of:
TBD.

13 Acknowledgements

TBD.

14 References

TBD.

14.1 Normative References

TBD.

14.2 Informative References

TBD.

15 Normative XSD and WSDL Specifications

TBD.

16 Issues

16.1 Minor, editorial documentation-related issues

- Resolve references
- Add CICS / DB2- type parallel Sysplex use cases
- Add some Logging References
- ? Block -> message
- cbe:CommonBaseEvent -> ogsa:CommonBaseEvent

16.2 More significant documentation-related issues

- Morph service data into createServiceExtensibility expressions
- Add operation exceptions
- Rename attributes and portTypes so as to be less MVS System Logger-centric.
- Isolate OS-specific attributes and permit them to default to LogManager-resident, startup-time-specified values.
- Turn notes for operations into xml descriptions.
- Turn UML representation of XML into XML text.
- Add (enf-like) events generated by Logger Subsystem
- Support for general plugins.
- DB … “guarantee that power outages or system crashes do not corrupt the log”

16.3 Technical issues

- Do we accommodate MODE= SYNC / SYNCECB / ASYNC?
- Should ehlq / hlq be exposed for all OSs?
- What is our relationship to policy?