GOC-TX: A Reliable Ticket Synchronization Application for the Open Science Grid

Soichi Hayashi
Indiana University
2719 E 10th Street
Bloomington, IN 47401
+1-812- 606-7104
hayashis@indiana.edu

Arvind Gopu
Indiana University
2719 E 10th Street
Bloomington, IN 47401
+1-812-856-0187
agopu@indiana.edu

Robert Quick
Indiana University
535 W Michigan Street
Indianapolis, IN 46202
+1-317-274-5260
rquick@iu.edu

ABSTRACT
One of the major operational issues faced by large multi-institutional collaborations is permitting its users and support staff to use their native ticket tracking environment while also exchanging these tickets with collaborators. After several failed attempts at email-parser based ticket exchanges, the OSG Operations Group has designed a comprehensive ticket synchronizing application. The GOC-TX application uses web-service interfaces offered by various commercial, open source and other homegrown ticketing systems, to synchronize tickets between two or more of these systems. GOC-TX operates independently from any ticketing system. It can be triggered by one ticketing system via email, active messaging, or a web-service call to check for current sync-status, pull applicable recent updates since prior synchronizations to the source ticket, and apply the updates to a destination ticket. The currently deployed production version of GOC-TX is able to synchronize tickets between the Numara Footprints ticketing system used by the OSG and the following systems: European Grid Initiative’s system Global Grid User Support (GGUS) and the Request Tracker (RT) system used by Brookhaven. Additional interfaces to the BMC Remedy system used by Fermilab, and to other instances of RT used by other OSG partners, are expected to be completed in summer 2010. A fully configurable open source version is expected to be made available by early autumn 2010. This paper will cover the structure of the GOC-TX application, its evolution, and the problems encountered by OSG Operations group with ticket exchange within the OSG Collaboration.

General Terms
Algorithms, Design, Reliability, Experimentation, Standardization.

Keywords
Ticketing, Synchronization, Algorithm, Grid Computing, Collaboration, Support Center

1. INTRODUCTION
Even as the Open Science Grid [1] was being conceptualized, the idea of providing support across institutional barriers was a concern. Individuals from the several organizations needed to be able to access and act on trouble tickets. This support could be done more efficiently if the individuals providing that support did not have to access an offsite ticketing service with its own policy and procedures, user interface, reporting mechanisms and authorization and authentication standards.

To allow this, OSG Operations [2] undertook the project of exchanging ticket information between several ticketing services. Initial attempts were done using email based ticketing procedures on each local instance. This became unreliable quickly due to several factors including: updates to the schema at any location affecting all participants, institutional spam filtering, unreliable email parsing, and general lack of standards email formats for this purpose. In several cases the department that ran the ticketing service was also removed from the department responsible for OSG support, complicating communication lines even further. Overall email based exchanging of tickets became too expensive and unreliable for OSG Operations to provide with any quality assurance.

At this point OSG Operations decided to work on interfacing ticketing systems to synchronize data via web services, which several commercial and homegrown ticketing services provide these days. This paper provides detailed information on the design and implementation of GOC-TX, a generic Ticket Synchronizer [5], and the outcome of the work done at Indiana University to move this service into production within the context of the OSG.

2. TICKET EXCHANGE – THE BASICS
2.1 Definition of Ticket-Exchange
Most large collaborations, including grid computing projects like the Open Science Grid (OSG), and the TeraGrid [3], often consist of multiple institutions, and/or several departments within an institution. These individual member institutions or departments often have their own ticketing system to track issues, bugs, feature-requests, etc. Additionally, there is sometimes a central traffic-cop type entity that coordinates the overall operations of such collaborations. The Grid Operations group for the Open Science Grid is an example of one such entity.

In certain collaborations the participating entities may agree to use one ticketing system, perhaps offered by the central coordinating agency; Fermilab’s use of the Remedy ticket system [11] for all its divisions, and Indiana University’s use of the Footprints system for its campus Information Technology support needs are two examples. However, more often than not, the individual collaborators prefer that their users and support staff be allowed to

Published under licence by IOP Publishing Ltd
continue using that ticketing system, while also exchanging these tickets with the rest of the collaboration. Examples are the support centers associated with the Open Science Grid (OSG), the constituent resource providers of the TeraGrid, and various constituent groups in the European Grid Initiative [6]. This gives rise to the need for a reliable ticket-exchange setup.

Traditional ticket-exchange setups have tended to be based on emails from one ticketing system to another; and custom parsers to enable exchange of ticket information. This is often very error-prone. The OSG Operations group has designed and developed an alternative web-services based ticket-exchange system. The following two sections will explain more about traditional email-based setups; as well as introduce GOC-TX in more detail.

2.2 Email-Based Ticket Exchange

Until the end of 2009, OSG Operations used an email-parsing based system to exchange and process tickets between different ticketing systems used by various OSG support centers. We will use their setup as an example to describe the most commonly used email-based ticket exchange setup. See Error! Reference source not found. for a schematic of this.

The sequence of events shown on Error! Reference source not found. is as follows:

1. OSG user opens a ticket using a web submission form on the GOCTicket web site [4] – an OSG Operations Support staff member is alerted via various mechanisms – or by sending email to the OSG Operations Support group. The GOCTicket website uses a Footprints ticket system as its back-end.
2. Depending on user selections, the ticket is assigned automatically or by support staff to “Support Center A” within the Footprints ticketing system for further processing.
   a. Footprints constructs a notification email using its proprietary algorithm and preset templates, and sends it to support center A's ticketing system (T2).
3. Ticket system A uses its own proprietary algorithm, and parses the email for meta-data and content.
   a. Ticket system A uses the meta-data to decide if this is an existing local ticket. It creates a new ticket, or updates an existing ticket.

2.3 Web-Services Based Ticket Exchange

See Error! Reference source not found. for a schematic of the web-services based design used by the GOC-TX application.

The sequence of events shown in Error! Reference source not found. is as follows:

1. OSG User opens a ticket using a web submission form on the GOCTicket web site, or by sending email to the OSG Operations Support group – in which case OSG Operations Support staff will create a ticket on the user’s behalf. The GOCTicket website uses a Footprints ticket system as its back-end.
2. Depending on user selections, the ticket is assigned automatically or by support staff to Ticket Exchanger A (referred to as “TX-A” from now on) within the Footprints ticketing system.
3. A trigger is sent to GOC TX indicating the two ticketing systems involved – OSG Footprints and ticket system A – including the source ticket ID.
4. GOC-TX receives the trigger, and executes synchronization between source and destination ticket systems.
5. GOC-TX uses a pre-configured Footprints-API to get source ticket details; then it converts the contents into a format understood by the destination ticket system A.
6. GOC-TX uses a pre-configured ticket system A-API to create a new ticket or update an existing ticket on the destination ticket system.
7. Support center A processes this ticket per their guidelines, and updates the ticket with periodic updates within ticket system A.
8. When support center A updates their ticket, a trigger is sent to GOC TX indicating the two ticketing systems involved – ticket system A and OSG Footprints – including the source ticket ID.
9. GOC-TX repeats steps 4, 5, and 6, in reverse direction.
10. An OSG Operations staff member and the user get notified about various updates only from the central GOCTicket system.

### 2.4 Comparison Between Email-Based and GOC-TX-Based Ticket Exchange

In Error! Reference source not found. below, we list some of the major technical challenges associated with the traditional email-based approach, and how GOC-TX deals with them.

**Table 1: Comparing ticket exchange based on email and GOC-TX**

| Email-based System | GOC-TX-based System |
|--------------------|---------------------|
| Each time an email is sent to an email parser, the destination ticketing systems often reply with an acknowledgement email, and this often creates an infinite email loop. Ticketing systems are tweaked to prevent this by configuring a list of email addresses to either not send the acknowledgement email, or to filter out and bypass normal processing for such emails from a certain list of email addresses. | Apart from parsing the ticket id from a trigger email (if email-based triggers are used), there is no other email parsing or acknowledgement and consequently no potential for ticket looping. |
| Often proprietary system interacts with another proprietary system; this makes it very hard to debug or fix any ticket exchange related issues or to add customizations needed to efficiently serve OSG community. | Conversion logic within GOC-TX can be customized easily. |
| Email templates, parsers, and email address used to communicate between the ticketing systems need to be highly static in order for this system to work. This makes it very hard to upgrade any ticket system without breaking ticket exchange. | The use of provider-published API creates resilience to ticket system upgrades, or configuration changes. |
| No central entity that monitors the entire flow of ticket exchange, thus making it difficult to detect, or troubleshoot issues. | Any issues related to ticket exchange can be closely monitored. |
| Ticket exchange is often only possible for textual content of tickets; it is hard to exchange metadata – for example, ticket type, priority, status, and other important information about each ticket, since that makes the parser even more complicated and hard to maintain. | Defined ticket conversion logic between any two ticket systems allows most types of information to be exchanged. |
| Every entity that wants to exchange tickets with another system must host, configure, and maintain their own exchange system. | GOC-TX can be hosted by one entity (for example OSG Operations) while other entities can easily use it for tickets. |
| Setting up email-based exchange often takes many iterations of trial and error until the details are worked out, even if a | If an entity uses a GOC-TX supported ticket system, it only |

### 3. GOC-TX

#### 3.1 Technical Components

A schematic of the various components that make up the GOC-TX application is shown in Error! Reference source not found. The major components are explained in further detail in the sub-sections below.

**3.1.1 Ticket Exchanger**

The Ticket Exchanger is the main driver of the application that accesses abstract accessors and converters – explained in sections 3.1.2 and 3.1.3 – to do the actual synchronization of tickets.

**Figure 3: GOC-TX Component Schematic**

#### 3.1.2 Accessor

Accessors provide the ticket exchanger a means to access various ticketing systems, by providing the following interfaces:

```java
public interface TicketAccessor {
    public interface TicketAccessor {
        public String create(Ticket ticket, String reverse_email);
        public void update(Ticket ticket, Date last_syntime);
        public Ticket get(String ticket_id);
    }
}
```

Concrete successors implement these interfaces by using the most appropriate means of communication to each ticketing system. [8] and Footprints [7] currently use SOAP [12], while RT [10] uses a RESTful interface [13].

**3.1.3 Converter**

A converter is takes two instances of the Ticket class (source and destination), and performs the conversion between the 2 tickets at the individual field level.
3.1.4 Trigger
A trigger is what causes GOC-TX to start the ticket synchronization between 2 tickets. Currently, we use email based trigger with a SMTP header that looks like following.

Delivered-To: tx+system1_system2@tx-itb.grid.iu.edu
Subject: ISSUE_ID=12345

SMTP headers and other content on the email are ignored. The email address "tx+system1_system2@tx-itb.grid.iu.edu" uses address extension protocol, "system1_system2" in this case, to pass a primary key type ID named TX-ID – a name that is mapped to a configuration file for specific set of accessors and converters used by the appropriate factory (explained in section 3.1.6). This ID identifies the ticket exchange a particular trigger attempting to fire. The example shown above shows a ticket exchange from one system to another.

The subject of the email is parsed by each ticket accessor's custom logic to pull the source ticket ID. More information on triggers is available in section 3.5.

3.1.5 Sync Table
The Sync table contains a list of source and destination ticket ID pairs that have been previously exchanged. This table is used to determine if the Ticket Exchanger needs to create a new ticket on the destination ticketing system or update existing ticket. It also contains the source ticketing system's last time stamp of synchronization, which is used to determine the parts of the ticket that needs to be synchronized.

The Sync Table includes the following columns:
- source_id (primary key)
- dest_id (primary key)
- tx_id (primary key)
- source_timestamp

At the time a ticket exchange is attempted, the absence of a sync record means that this is the very first exchange for the corresponding source ticket; this tells the Ticket Exchanger to create a new ticket on the destination ticketing system.

3.1.6 Factory
Ticket Exchanger passes a TX-ID from the trigger to a Factory (class instance) which then instantiates instances of required accessors, a converter, and initializes them with parameters specified in the appropriate configuration.

3.1.7 Ticket
This component, as the name indicates, is used to hold ticket information specific to each ticketing system -- usually a collection of get and set functions.

All classes inheriting this abstract component class must contain at least following fields:
- Ticket ID
- Last UpdateTime.

3.2 Ticket Update/Creation Process - Sample Event Flow
The flowchart in Error! Reference source not found. shows a sample TX event flow for exchanging ticket from GGUS to Footprints.

We had to consider the following special cases during the ticket update / create process:
- By the time GOC-TX receives a trigger for an update, another user may have already added one or more additional updates to the source ticket. We need to pull all recent ticket updates since the last synchronization attempt updated destination ticket(s) in a ticket exchange set.
- When GOC-TX receives a trigger, destination ticket (as well as the source ticket that this trigger belongs to) may have already being updated. We need to check the timestamp on destination ticket as well the source ticket prior to updating the destination ticket. If destination ticket has been updated since the previous synchronization, then the following additional step must be done:
  - After GOC-TX has pulled the last ticket information from the source ticket, it should synchronize in reverse direction i.e. destination to source synchronization prior to proceeding with updating the destination ticket. After the source ticket is updated, GOC-TX needs to reset its metadata (status, priority, etc.) with the metadata which was just synchronized to destination in order for the two tickets to contain consistent metadata.
- GOC-TX may receive a trigger for a ticket that has already being synchronized in terms of content. This condition is often caused by a trigger email sent by a ticketing system acknowledging a previous synchronization of content made by GOC-TX itself — GOC-TX ignores such triggers unlike the ticket looping that results in the case of similar redundant acknowledgments in email-based ticket exchange.

Because of these special conditions, the actual implementation of ticket update / creation within Ticket Exchanger driver is more complicated than the schematic shown in Figure 3.
3.3 Customization
One of the key factors for creating a fully functional ticket exchange service was to be able to meet various demands from users without sacrificing the overall stability of the application. Our generic accessor and converter for GGUS, RT, or FP ticketing system would work for most cases but we almost always need to implement custom logic and features specific to each support center.

For example, one institution may require that certain custom fields be populated based on metadata stored on the source ticketing system. Another may require that such metadata be added to ticket description instead of being stored as custom field.

In order to accomplish this, we have adopted an Object Oriented approach to our design of GOC-TX's generic accessors and converters. These components are designed to be easily extended to more specific accessors, converters, etc. as necessary. For example, we have created a BNL-RT accessor that extends the generic RT accessor, and a BNL2FP converter that extends the generic RT2FP converter to provide a functional ticket exchange with Brookhaven National Laboratories (BNL) [9], which currently uses an RT ticketing system with custom configuration. The GOC-TX factory class is able to then instantiate these derived classes as configured for RT2GOC TX-ID using a plug-in architecture as shown in Figure 5.

Centralizing the ticket exchange system, and handling all of the business logic within a single application allows us to implement features such as this easily, something we were unable to do previously with email-based ticket exchange.

3.4 Synchronization
GOC-TX relies on a source ticket's last synchronized timestamp, stored in the sync table, in order to figure out the recent parts of a source ticket that need to be synchronized. The use of source timestamp also eliminates the possibility of occurrence of ticket loops similar to acknowledgment email loop observed in an email based ticket exchange system.

Figure 6 shows a typical list of transactions during a ticket update. Note that GOC-TX never uses its own server's timestamp because each ticketing system may contain slightly different timestamp from what its server uses.

This is also important since most ticketing system may not update the "last update date" right after GOC-TX commits an update. For each ticket update, GOC-TX must pull ticket information from the destination ticket immediately after the update, and store it to the sync table and store it as that ticket system's last synchronized timestamp.

The sequence of events shown on Error! Reference source not found. is as follows:

1. Initial State
2. Someone updates ticket 123
3. GOC-TX Actions
   a. GOC-TX receives a trigger from Support Center A
   b. GOC-TX retrieves last update time for ticket 123. (12:00 in example)
   c. GOC-TX retrieves last update time from sync table between ticket 123 and 456. (11:00 in example)
   d. Since the timestamp is more recent, it proceeds with synchronization

   GOC-TX Sync Table
   Ticket 123 @ Support Center A
   Ticket 456 @ Support Center B
   123 (11:00) 456
   456 (11:01) 123
   Last Update: 11:00
   Last Update: 11:05

   GOC-TX updates destination ticket 456 with changes made after 11:00 on ticket 123. The update time for ticket 456 is updated by ticketing system B.

   GOC-TX updates sync table for the 123-456 link with the new timestamp.

   GOC-TX retrieves new "last update time" for ticket 456. (12:03 in example), and update sync table for the reverse link.

   When GOC-TX detects that there are more than one updates made since last synchronization, it attempts to combine all new updates into one update. This behavior is implemented at accessor level as a convention, so each accessor can choose to insert individual updates separately or group them together, depending on particular exchange requirements.

3.5 More on Triggers
As stated earlier, trigger is the method used by GOC-TX to initiate actual ticket synchronization. We currently use email-based triggers, while an ActiveMQ [14] based trigger system is in development. In this section, we explain the most obvious issues associated with email based triggers, and our current mitigation process for each of those issues.

Issue 1: Email delivery could fail, or GOC-TX server maybe temporarily down so that GOC-TX does not receive a trigger.
Mitigation: SMTP server usually can be configured so that if delivery of an email fails, it will retry sending to the destination SMTP server. As long as one of the trigger emails gets to GOC-TX, GOC-TX will be able to synchronize the ticket with all updates that occurred since the previous update.

Issue 2: GOC-TX could fail to initialize and ignores trigger even if it was delivered to GOC-TX mail server successfully

Mitigation: Monitor our mail server and GOC-TX bootstrap setup so that this condition can be detected, and sent as an alert to the GOC-TX administrators.

This is the main reason why we are already implementing an ActiveMQ based trigger interface.

Issue 3: GOC TX needs to parse the email subject to pull ticket ID.

Mitigation: The parser can be configured as a plug-in. Most ticketing systems allow customizing of the subject line so that only Ticket ID itself is present within the subject line. If this is the case, then practically no parsing is necessary (subject = ticket ID). Also, any parsing issue can be monitored closely.

Currently, none of the above issues have proven to be serious enough, to outweigh the main benefit of email-based triggering – providing a very easy way for OSG support centers to start using GOC-TX.

4. Advanced Ticket Exchange Topics

4.1 Dealing with multiple TX links

In previous sections, we have explained ticket synchronization between only two ticketing systems. GOC-TX can also handle ticket exchange that is setup so that it has more than two ticketing systems. Figure 8 shows a sample ticket 456 having two TX links.

![Figure 8: Ticket Exchange Involving Three Ticket Systems](image)

GOC-TX can cascade ticket updates originating on one ticket. It can also deal with synchronizing in the case where updates are made simultaneously on all three tickets regardless of the order of arrival of triggers. GOC-TX handles cascading updates correctly because at any given time instant, it only sees the update as an update between two tickets. Using the example in Figure 8, if someone updates ticket 123 which then causes ticket 456 to be updated (link A). In TX link B, the update occurs simply because of the update to ticket 456, thereby cascading the update to ticket 789. From GOC-TX’s point of view, it does not matter which ticket system updates the ticket or when.

![Figure 9: Ticket Exchange Involving Three Ticket Systems (Two within one Support Center)](image)

The fact that GOC-TX acts as a ticket broker that works independently of any support center or central group, means a ticket does not necessarily have to be exchanged via OSG Operations. This allows providing a ticket exchange service without the involvement of OSG Operations, if the need for direct exchange between two collaborators arises. In the example shown in Figure 9, Support Center B has two ticketing systems. It can use the same instance of GOC-TX used to exchange tickets on link A-B to also enable internal ticket exchange between systems B and C. GOC-TX thus could be used as a generic ticket exchange service that meets the needs of various support centers.

GOC-TX, however, will not be able to synchronize if tickets are exchanged in a loop, for example, as shown in Figure 10.

![Figure 10: Example Unsupported Ticket Exchange Setup](image)

In this case, the update on ticket 123 will initiate clockwise and counter-clockwise ticket updates and it will cause infinite update loop. So far, our topology is setup so that a requirement like this cannot be satisfied. We can implement a mechanism to prevent infinite looping in the future.

4.2 Monitoring

One of the key motivations for the development of GOC-TX was that it was very difficult to detect and diagnose any issues within traditional email-based ticket exchange. For example, if a ticket update did not happen between two ticketing systems, it could be because of any of following causes:

- Source ticket system did not send out the expected notification.
- Source ticket system sent the notification to a wrong email address.
- Destination ticket system did not receive the expected email.
- Destination ticket system wrongly filtered out an email from the source ticketing system.
- Destination ticket system had errors processing the notification.

Since both source and destination ticketing systems are outside the control realm of any central coordinating group like the OSG Operations group, several parties need to monitor various components on their end without knowing if other components are causing problems. Since each support centers works independently for the most part, unexpected issues occur too often, and make it very difficult to maintain a stable ticket exchange service.

GOC-TX collates the various parts of ticket exchange in one location, which allows a centrally located group like OSG Operations to monitor, and get alerted by any error or warning conditions immediately after they has been discovered. Fixing these issues is an easier task since one entity has full control over the GOC-TX application.

GOC-TX also allows OSG Operations test and simulate various test-cases before releasing it to production system; this was very
difficult to do with the email based ticket exchange system since not all support centers have a test instance of their ticketing system to participate in such tests. GOC-TX allows OSG the operations group to simulate ticket exchange while monitoring the entire exchange process, using test instances of various ticketing systems at Indiana University.

5. CONCLUSION
The GOC-TX application uses web-service interfaces to solve one of the major operational issues faced by large collaborations, including grid computing projects like the Open Science Grid (OSG), and the TeraGrid. It helps these collaborations allow their users and support staff to use their home ticketing service environment while also exchanging these tickets with collaborating institutions. It operates independently from any ticketing system, and avoids several problems associated with email-based ticket exchange systems. It uses triggers – email, active messaging, or a web-services call – from one ticketing system to check for synchronization status, and is able to apply the recent updates to destination ticket(s).

The currently deployed production version of GOC-TX is able to synchronize tickets between the Numara Footprints ticketing system used by the OSG and the following systems: European Grid Initiative’s system Global Grid User Support (GGUS) and the Request Tracker (RT) system used by Brookhaven. These services have run reliably for several months and other than regular maintenance have not experienced the issues previously encountered by the email-based exchange, thus decreasing the load on OSG Operations and the people supporting our users.

6. FUTURE WORK
OSG Operations will continue to work with OSG collaborators to eliminate legacy email-based exchanges. This will include a Remedy based interface and more RT based interfaces. Once OSG is entirely updated to use GOC-TX, we will continue standard maintenance on all interfaces as ticket services evolve, and local policy and schema at the hosting institutions change.

We expect to implement an additional ActiveMQ based trigger system. The proposed design would allow support centers to implement a mechanism, say a script invoked by their ticketing system when a ticket on their system is updated, to generate an ActiveMQ message that is programmatically readable; the mechanism could use JMS, Stomp, or any protocol that is most convenient to the support center to send the message to a central MQ Server. Our proposed design will also allow support centers, if they so wish, to continue to use email-based triggers but these email triggers after arrival within GOC-TX will be converted to MQ triggers, and ultimately stored on a central MQ server. An advanced support center could setup a more complex setup in which their MQ messages are first sent to a local MQ server, before being eventually delivered to the central MQ server using MQ’s NetworkConnector “store & forward” method.

We are also planning to release GOC-TX as an Open Source project so that other support centers within the OSG community, and other communities, can install, configure, and use it for their ticket exchange needs. We hope that exposing GOC-TX to a larger group of users will allow it to become a more stable and secure application, will allow the Open Source community to contribute more accessors, converters, etc. for various other ticketing systems.

7. ACKNOWLEDGMENTS
Our thanks to Guenter Grein of EGI, and Jason Smith from BNL, for testing assistance during the initial development phase.

8. REFERENCES
[1] Open Science Grid (OSG) home: http://www.opensciencegrid.org/
[2] OSG Operations home: http://osgoc.blogspot.com/
[3] TeraGrid home: https://www.teragrid.org/
[4] GOCTicket - Central ticketing system for the OSG: https://ticket.grid.iu.edu/goc/
[5] GOC-TX Development home: http://sites.google.com/site/osggocdev/projects/goc-tx
[6] European Grid Initiative (EGI) home: http://www.egi.eu/
[7] Numara Footprints software home: http://www.numarasoftware.com/FootPrints/service_desk_software.aspx
[8] GGUS ticket system home: https://gus.fzk.de/
[9] Brookhaven National Laboratories (BNL) home: http://www.bnl.gov
[10] Request Tracker (RT) ticket system home: http://bestpractical.com/rt/
[11] BMC Remedy ticket system home: http://www.remedy.com
[12] Simple Object Access Protocol (SOAP) on Wikipedia: http://en.wikipedia.org/wiki/SOAP
[13] RESTful Interface on Wikipedia: http://en.wikipedia.org/wiki/Representational_State_Transfer
[14] ActiveMQ home: http://activemq.apache.org/>