### ABSTRACT

This document reports on the implementation of the technical infrastructure to support the community of teachers using tools within the ITEC project. This infrastructure description is referred to broadly as the ‘information model’. The model is presented in four main views: 1. The technical architecture to support community-led curation of tools; 2. The design and implementation of new tools and the facilitation of additions to tools; 3. The requirements for ongoing maintenance and management of a large-scale Educational Tool Repository.

### AUTHOR, COMPANY

Dai Griffiths, Mark Johnson, Kris Popat, University of Bolton

### WORKPACKAGE

WP 08

### CONTRACT NO

2577566

### DATE

31/08/2011

---

1

PU = Public

PP = Restricted to other programme participants (including the EC services);

RE = Restricted to a group specified by the Consortium (including the EC services);

CO = Confidential, only for members of the Consortium (including the EC services);

INN - Internal only, only the members of the consortium (excluding the EC services)
<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Reason of change</th>
<th>Status</th>
<th>Distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>V1</td>
<td>15/9/2012</td>
<td>1st Draft</td>
<td>Draft</td>
<td>University of Bolton</td>
</tr>
<tr>
<td>V2</td>
<td><code>08/9/2012</code></td>
<td>Submitted version</td>
<td>Official</td>
<td>University of Bolton</td>
</tr>
</tbody>
</table>

FILING CODE: ITEC-D8.2_v2.doc

RELATED ITEMS:

DOCUMENT HISTORY
Executive summary

In 2011/12, Work Package 8 has progressed from investigations around tools and services establishing “Mashup connectors” to realising the technical infrastructure upon which the coalface delivery of iTEC scenarios is based. The information environment within which tools work has been incrementally determined and codified in the form of working tools, data flows, use-cases and dynamic procedures to assist in the easy and manageable evolution of these tools.

Having decided on a W3C widget approach, WP8 has delivered a fully-functional ‘widget-store’ for the curation of educational tools which can be easily instantiated by teachers. The information flows around the widget store have been designed so as to facilitate cross-community engagement by teachers, learners, ministry officials and education managers. These are documented in this report in the form of descriptions of tools, use-cases and the processes by which enhancements have been developed. The iTEC Widget Store and associated software constitute ID8.4 – Widget Server ‘community model’ incorporating collaboration, commenting, tagging and contributions from teachers. As required in the DOW, this provides a community base for innovation in schools across the project and extends efforts to drive teacher innovation in the use and creation of widgets. All the code for these applications is being developed using an open methodology, and largely contributes to existing open source projects (particularly Apache Wookie and Edukapp) in order to maximise the effective use of project resources and the prospects for valorisation.

The information model for the technical support of active communities of teachers has a number of dimensions which have led to the formulation of particular strategic priorities. These are shown in the table below, together with the actions taken to address them.

<table>
<thead>
<tr>
<th>Strategic Priority</th>
<th>WP 8 Achievement (with reference to chapter headings)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resolve technical challenges in embedding services across a range of platforms.</td>
<td>The full Wookie API has been implemented in the Widget Store. Mechanisms for widget creation, and integration with physical devices have been investigated. Interoperability with W3C widgets and paradata has been maintained. See Sections 2 and 3</td>
</tr>
<tr>
<td>Provide a simple means whereby teachers can include services and resources within their teaching activities</td>
<td>Implementation of the Widget Store as an embeddable component, providing access to widgets. See Section 2</td>
</tr>
<tr>
<td>Enable teachers to contribute and creating widgets.</td>
<td>Upload capabilities have been included in the Widget store. These can upload existing widget files. They can also enable teachers to create simple widgets by ‘wrapping’ web resources and services. <strong>See Section 3.2</strong></td>
</tr>
<tr>
<td>Support teachers in finding the services and resources they need.</td>
<td>Rich description mechanisms have been provided for widgets on the repository. These are in line with the iTEC taxonomy from WP10. Enhanced metadata capabilities have been included, and new paradata capabilities introduced. <strong>See Section 2.6</strong></td>
</tr>
</tbody>
</table>

These achievements map on to the detailed tasks of the workplan, as shown on page 10.
# TABLE OF CONTENTS

Reminder of the context .................................................................................................................. 8
Purpose and scope of the task ......................................................................................................... 8
Relationship with other tasks .......................................................................................................... 9

1. Impacts of Work Package 8 ........................................................................................................ 10
   1.1 Risk Analysis Review ........................................................................................................ 12
   1.2 Ethical Issues ................................................................................................................... 14
   1.3 IPR issues ......................................................................................................................... 14

2. Extending the Wookie information model to support community engagement ...................... 15
   2.1 Technical Developments in the Evolution of Apache Wookie ........................................ 15
   2.2 Implementation of the full Wookie API .......................................................................... 17
   2.3 Open Authentication Integration .................................................................................... 18
   2.4 Development of the store ............................................................................................... 19
   2.5 Social paradata ................................................................................................................ 20
   2.6 Describing widget functionalities .................................................................................... 21
   2.7 User stories for the iTEC Widget Store ........................................................................... 21
   2.8 The Widget Store service based upon Edukapp ............................................................. 27
   2.9 Extending the store API to describe widgets with social metadata and paradata .... 28
   2.10 Interfaces with other iTEC services ............................................................................. 30
   2.11 Preparation for pilots ...................................................................................................... 32

3. Innovation and enrichment of the mashup shell ....................................................................... 34
   3.1 Integration of Physical Devices with the Widget Architecture ....................................... 35
   3.2 Import features of the store for easy creation of widgets ................................................. 37
   3.3 Widget creation tools assessed ....................................................................................... 39
   3.4 Rationale behind widget development ............................................................................ 40
3.5 The iTEC widget development strategy ................................................................. 40
3.6 Widgets providing iTEC cloud interfaces ............................................................... 42
4. Maintenance and Server Management .................................................................... 44
   Impact on pre-standardisation ............................................................................. 46
5. Conclusion .................................................................................................................. 47
Appendices ................................................................................................................... 48
Appendix I: Table of Widget Developments ............................................................... 48
Appendix II: Phase Two Widget Developments ......................................................... 51
Appendix III: Phase Three Widget Developments ..................................................... 52
Appendix IV Store REST Api ..................................................................................... 53
Reminder of the context

The context of Work Package 8 as stated in D8.1 is:

“ITEC seeks to transform teaching practice in schools through the development and implementation of innovative pedagogical scenarios which transcend the organisational barriers of existing learning technologies. Principal amongst these barriers is the interoperability between tools across different learning platforms and in the different contexts within which learners learn. Work Package 8 is principally concerned with these issues of interoperability and the consequent needs for tool description and discoverability in the ‘classroom of the future’ where educational online tools exist independently of learning platforms. Work Package 8’s technical solution is to implement an Educational Application Store based on the W3C widget specification and the Apache Wookie Widget Server. Most broadly however, this technical work can be situated as a continuation of the efforts within Learning Technology to establish the means for increasing personalisation and technological flexibility in education.”

Here we report on the progress made in delivering the Educational Application Store (Widget Store) for the purposes of providing a platform for nurturing and sustaining innovation by teachers both within and outside the context of ITEC scenarios.

Purpose and scope of the task

The deliverable D8.2 is described as follows:

Building on the description framework established in year 1, the information model report will summarise the specific iTEC components, their fit with iTEC scenarios and the information flows within the educational community through engagement with iTEC technologies. The efficacy of the information model will be demonstrated by new technological procedures for user-driven instantiation of iTEC components in scenario contexts, user-driven creation of new components, user collaboration, sharing of practice and technologies, and new components that meet or extend scenario possibilities.

The principal component in meeting this task has been the creation of a technical platform for the support of the education community in the realisation of scenarios. Whilst the most visible manifestation of this support is the ITEC widget store, much of the task has involved behind-the-scenes work on interoperability and architectural transformations so as to ensure that this support can be provided at large scale in the context of the broader goals of the project, and addressing the potential risks which naturally pertain to such a venture.
Relationship with other tasks

The work in creating the Widget Store has been done in collaboration with a number of other Workpackages and tasks. These are:

a. The contribution to the ITEC definition of innovation with Work Package 2 and the contribution to the scenario generation process. The latter has been significant since technical input into scenario generation has been important as a way of bridging the gap between technical and pedagogical work packages.

b. The support of scenario learning activities with work package 3

c. The engagement with teachers, both at a small scale and on a wider scale in conjunction with work package 6. Again this has helped to bridge the gap between technical activities and teachers.

d. The integration of the technical architecture of the Widget Store with the User Management and Access Control.

e. The provision of tools for the tagging of functionalities of widgets as will be consumed by the Scenario Development Environment in Work Package 10

f. The provision of infrastructure for other tools, including the composer (work package 7), and tools from Work Package 3 (e.g. TeamUp)
1. IMPACTS OF WORK PACKAGE 8

Milestone and Task Review

In the course of project work it has become clear that the critical intervention made by WP8 is (in the words of the D8.2 description) to support “tagging and discoverability of Widgets” in order to “create a space for early adoption and experimentation amongst teachers”. In order to provide a comprehensible system, rather than a collection of functionalities, this work has been focused around the development of the iTEC Widget App Store, which will be the interface for all interactions between teachers and the available services, and the context within which they can curate their own, or shared, collections of services and resources.

As a consequence of this project strategy, WP8 has prioritized work on Wookie and associated technologies in order to demonstrate the information model by developing new technological procedures for user-driven instantiation of iTEC components, as specified by the D8.2 deliverable description. The efficacy of the information model will be demonstrated by new technological procedures for user-driven instantiation of iTEC components in scenario contexts, user-driven creation of new components, user collaboration, sharing of practice and technologies, and new components that meet or extend scenario possibilities. All of these will be focused around, and facilitated by, the iTEC Widget Store.

In the executive summary to this document Table 1 shows how the work carried out addresses the strategic priorities established by the project. In carrying out this work, however, the tasks assigned to WP8 have been respected, with the Widget Store providing a focus for their activities. The following table details the work carried out within each task.

<table>
<thead>
<tr>
<th>Task No.</th>
<th>Short description</th>
<th>Achievements</th>
</tr>
</thead>
</table>
| T8.1 (m 1-36) | Investigation into tools/services and plug-ins of interest | a) Tools services and plug-ins have been developed including enriched interfaces with new means of user interaction
b) Extensive work has been carried out to integrate new widget-based tools for inclusion in Wookie, in particular work on the Widget Store functionality. See section 2 |
| T8.2 | Development of Mashup Connectors | This task was completed in month 12 |
| T8.3 | Development of missing | A strategy has been put in place to manage requests for additional functionality, and to coordinate development work. |
Work has been carried out with Arduino/JeeNode wireless development kit, and NodeJS to explore potential use in iTEC. Widgets have been developed to support Flash upload, and to facilitate the embedding of remote widgets.  

See section 3

This task has been the principal focus of work in the period documented by this deliverable (m12-24). It has involved the refactoring of the Apache Wookie server so as to support the creation of an independent widget store, and the development of that store.  

See section 2

This process fed into the functional requirements of the iTEC widget store.  

See Section 2.11

A demonstration server has been maintained at partner EUN which makes available the current version of the Apache Wookie Server and Widget store. This provides the infrastructure whereby teachers and administrators can upload and manage their own widgets.  

See section 4

The modelling of tools by means of functionalities, as established by WP10, has been implemented in the iTEC Widget Store. End users can make use of this to order found widgets, show ratings etc. Information on use will be added in the next period of project work.

The information held on the available widgets, their descriptions and use is also exposed by the server, and can be interrogated by any authorised front end application. This functionality will be made available for pilots.

The richness of the available information has been enhanced making use of the outputs of the SPAWS, which syndicates usage data across multiple Wookie installations.  

See section 2.6

WP8 maintains close contact with W3C, the authors of the widget specification used by the project. The Apache Wookie Widget Server, and the iTEC Widget Store are reference implementation of the W3C widget specification. The project has also integrated IMS
1.1 Risk Analysis Review

The risks identified in the description of work which relate directly to the work undertaken for this deliverable concern:

1. **The exponential speed of innovation**

   Whilst ‘innovation’ has been a focal point of the project and Work Package 8 has been involved in work on what ‘innovation’ means in the context of iTEC (see section 3), early indications are that a the community-driven store which allows teachers to curate their own personal collections of educational tools will provide a means whereby teachers themselves can drive innovation.

2. **Too little interaction between the developers of scenarios and developers of technical platform.**

   Work Package 8’s involvement with the pilot user testing and direct engagement with teachers has been essential in ensuring that developments meet the particular needs of teachers. More direct user testing and greater technical autonomy for teachers through use of the Widget Store should ensure that developers continue to listen to those who trying to innovate in their classrooms.

3. **The integration of Wookie Widgets may be challenging in terms of some future classroom technologies.**

   ITEC is now largely focused on Widget technologies, and their simplicity and flexibility across a range of different platforms has been demonstrated. Having said this, some technologies which may be useful to education (e.g. Skype) do not easily fit into widgets. However, Work Package 8’s investigation into real-time widgets (see Section 3) and physical devices can provide ways of addressing some of these problems.

4. **Interoperation between particular Widgets may sometimes be problematic.**

   Widget interoperation has yet to show itself to be a major problem for the project. It may be that what appears as a technical requirement, in practice isn’t a priority with regard to innovative pedagogy. However, an increasing number of kinds of solutions to
interoperability are becoming available – particularly those using NodeJS and other real-time technologies.

5. **Use of some ‘Listener’ widgets may be impractical**

Listener and hardware integration is at an experimental stage but results are encouraging. Far from being impractical, they may provide new ways of creating technologically-enhanced shared experiences in the classroom.

6. **Stakeholders may choose not to engage in the Widget community server.**

This remains to be seen on a large scale, however, the ease with which pilot teachers were able to find widgets and embed them into the store, and their enthusiasm for doing this suggests that the Widget Store should be well-used as a personal repository for tools for teachers.
1.2 Ethical Issues

No ethical issues have arisen during the course of work for Work Package 8.

1.3 IPR issues

Wookie software developed in Work Package 8 is available under the GPL and made available through the Apache foundation.
2. EXTENDING THE Wookie INFORMATION MODEL TO SUPPORT COMMUNITY ENGAGEMENT

The Wookie service as it existed prior to the developments of ITEC was a simple technical solution to the injecting of tool functionality across a variety of technical contexts (e.g. learning systems, blogging platforms, etc). The rationale was simply “Why have bespoke toolsets for different platforms? Why not have a central toolbase from where tools can be instantiated across a variety of contexts?”

The requirements of iTEC have meant that on the one hand, a central toolbase is highly desirable. On the other hand, the experience of using a central toolbase with thousands of teachers across Europe is untried. How should it best be organised? Therefore, beyond addressing the technical requirements of providing a central toolbase (the “Widget store”), the project has required creating a technical platform and an associated information model for supporting a large community of teachers who will have different requirements, accessing the store from different platforms, wishing to contribute their own tools, or curating personal collections of tools they find effective in their classrooms. How can these issues of flexibility and scale be managed whilst maintaining interfaces for adding tools to learning platforms which are no more complex than adding ‘native’ tools?

Moreover, given the potential scale of the usage of a central Widget Store, the hope is that active teaching engagement with the store affords opportunities for tagging and commenting which in itself will bring the benefits of sharing practice amongst teachers across Europe. The management of a curated store of widgets as a social network also needs to be considered.

The Widget Store exists within an ecology of tools and services which make up the ITEC platform. Central to ITEC’s philosophy is the realisation of teaching and learning scenarios, and for this tools must be discoverable by the functionalities they provide so as to present themselves as possibilities to teachers or educational managers or designers. Maintaining this important link to the scenario philosophy of ITEC is also something that has had to be considered in the information model.

Finally, an ecology of tools and services like ITEC can present new complexities as users seek to move from one service to another: i.e. from the Widget Store to the Composer or to the People and Events Database. Like all the other components of ITEC, the Widget Store has been integrated with the single-sign-on technology provided through authentication with the ITEC cloud through the User Management and Access Control (UMAC) interfaces.

2.1 Technical Developments in the Evolution of Apache Wookie

During the last year extensive work has been carried out by WP8 to contribute to Apache Wookie so that it better supports the functionality required by iTEC. In this period Apache Wookie has undergone a number of release iterations and is now at version 0.11. Each release has enriched its
capabilities and resolved bugs while maintaining its compliance with the W3C specification as it has evolved.

A significant change to the model has been made in response to the need for a role to be represented which can describe the user associated with a widget instance. To clarify this it is necessary to outline the process whereby Wookie stores and presents widgets to a shell. A widget package is uploaded to Apache Wookie, is stored on the disk and unzipped, ready for its contents to be delivered to the shell. At the point where a shell requests a widget certain items of information are required from the shell to enable Wookie to set up and send an instance of that widget. A widget instance is a representation of a widget, which is geared for a particular context. It is this contextual information that the shell must send to Wookie before the instance can be generated. The contextual information prior to the model change, included the following:

- **An API Key.** The API key is a secret key registered within Wookie for a particular shell.
- **A sharedDataKey:** this is a key designed to be used within a particular context. This might be a page, or a course, allowing the widget instance to be unique to that context.
- **A User:** A minimal user includes a screen name and id allowing the user screen name to be displayed in a widget. The chat widget is a good example of a widget that uses this information.

The programmatic workflow used by a shell plug-in to set up a widget instance is described in this following diagram (Figure 1).

![Programmatic Process Diagram](image-url)

**Figure 1: Wookie Programmatic Process**
As can be seen the contextual information associated with a widget instance is geared towards the place where that instance is displayed, and not towards how it might be used. This is satisfactory for widgets that only need to display immediate information or which are designed for collaboration where no setup is required. However, this does not allow widgets to represent the different actions interactions which may need to be made available to different by different users. In the context of iTEC this is a limitation, as there is a potential difference between how a teacher may wish to use a widget and the patterns of use of a student. For example, the teacher might need to set up some information in a particular widget, customizing its purpose for a particular group or class. A voting widget, for instance, needs to be configured with the question or issue to be voted upon, a chat widget might need to be moderated, a test widget would need the question to be set up.

To address this, the concept of user role has been added to the user information. This role information can then be used by a widget to display itself accordingly. Wookie is agnostic where specifics of role information are required. It is a property that can be used flexibly for a given instance of wookie. In the instance of Wookie established by the project three predefined roles have been implemented.

- Administrator
- Teacher
- Student

This small set of roles encapsulates what is required by iTEC widgets at the level of Wookie. There are other services, however, which allow a finer grained approach to roles and permissions. These are defined within the open authentication integration work which has been carried out by WP7 as part of it’s user management and access control service.

### 2.2 Implementation of the full Wookie API

In the work reported in D8.1 the Wookie Application Programmer’s Interface was extended to cover all the major functional end-points of wookie. These included:

- Getting a widget list
- Getting an individual widget information
- Getting a widget instance
- Adding a participant to a widget instance
- Setting properties for a widget instance
- Adding widget files via upload
- Getting and creating security keys (API keys)
- Getting and creating policies
The full list and details of the interface can be found here (http://incubator.apache.org/wookie/docs/api.html)

This removed from Wookie a dependency on an outdated and inflexible administration interface. Before the API was created, the only way of adding widgets, setting security keys etc. was via the administration user interface which was a set of JSP pages built into Wookie.

In the work reported in this deliverable the Wookie connector framework used by the iTEC Widget Store has been extended to cover all aspects of the API. The original connector provided enough functionality to enable a plug-in developer to connect to Wookie, view the widget list and create an instance of a selected widget for display. The connector framework now covers the full API. This has allowed other clients to be created to carry out these administrative and functional activities, which are entirely separate from Wookie itself.

This was especially important for iTEC. Wookie is used to store the widgets used in the project, but the user experience of managing them was disjointed. Moreover additional features were required which were not available in the old Wookie management interface, in order to allow the iTEC community to use widgets in the way they wanted. Perhaps even more significant is the fact that upload and management of widgets can now be carried out remotely. This is important because it has allowed an extended widget store to be developed which not only handles widgets and their properties, but also seamlessly adds features for reviewing, rating and tagging widgets. It also has allowed other iTEC services which manage widgets to do so more completely such as the iTEC composer being developed in work package seven.

2.3 Open Authentication Integration

Work package seven has been responsible for the development of user management and access control or UMAC. This work has moved the project technical services closer to single sign-on and a single managed identity for the users. At the most basic level the user will be able to use the same credentials for accessing all the different services, the iTEC Composer, the People and Events Directory and the iTEC widget store.

More importantly it lays the foundation for creating a single access point to the services where the user only has to login once. This access point might be the shell in which they are working. Moodle for instance is one of the shells supported by iTEC. The user, whether they are a teacher or a student, will be able to log into moodle and simply navigate to the widgets store and other services without any need to understand the complexities of the underlying system.

This fully integrated solution was still in development at the time of writing this deliverable. The plan is to have full integration for the cycle 4 pilots. UMAC is available currently and each service integrates with it, so a single user identity is available now.

To enable this to happen work was done on Wookie itself within WP8 to enable it to connect directly with the UMAC implementation underneath. The details of this work are documented in D7.2.
Figure 2 above shows how the various access points to UMAC tie together the user identity in a seamless way. The arrows indicate flows of user information between the components. The intention is that log in is only required at the Shell level and all other services receive user credential information from the shell and then authenticate this information with UMAC. In the case of the Store it does this directly. In the case of Widgets this is done through the modifications made to Wookie. The way in which widgets need to be written in order to take advantage of this is described in D7.2. This method will be adopted, as appropriate (some widgets do not need user identity – a calculator for instance). This is outlined in the widget development strategy that is described later in this document.

2.4 Development of the store

Deliverable D8.1 set out a plan for the development of a widget store that went beyond the current capabilities of Wookie, and described its rationale as follows:

Addressing the interoperability barriers to using tools across environments is essential if any change in practice is to be made possible. However, practice change will not necessarily follow the removal of technical barriers. The need to encourage and inspire teachers with what is possible and with what becomes (following the removal of technical barriers) ‘ready-to-hand’ for them remains. Teachers are very busy, and often over-burdened with new systems and practices. Thus anything which iTEC does to engage teachers must in some way reduce their technological complexities rather than add to them.
It went on to describe the concept of an AppStore as a single place where tools of various kinds could be added in a seamless way. The commitment in D8.1 to the development of an AppStore is fulfilled in the present deliverable with the development and release of the iTEC Widget Store.

The removal of complexity is an important achievement of the Widget Store. However, it is important to be clear about the nature of this complexity. The teachers who use a store do not wish or need to know about the complexities of format, installation, technical requirements and similar aspects of widget deployment. This complexity should be hidden, and the process of adding some functionality to their shell or learning environment should be as simple and transparent as searching and choosing.

Nevertheless, there remains another aspect of complexity to be handled by the store. This concerns the criteria by means of which the store makes a decision to offer a particular tool. This cannot be hidden in the same way, and requires a suitable interface design. The tool needs to be categorized or labelled in some way. The categorizations can be described in purely functional terms, such as the functionalities of the tool. These functionalities could be aspects such as Video Tool, Chat client etc.. They can also be 'softer' categories such as Most Popular, Most Downloaded, Highest Rated. Thus WP8, having dealt with the principal underlying technical complexities of making the installation and use of tools cleaner and easier, is now addressing new set of complexities relating to choice and expression of views. The iTEC Widget Store is the means whereby these complexities can be addressed.

2.5 Social paradata

In the light of the previous section, we can see that many of the additions to the technical capabilities of Wookie reported in this deliverable are designed to address the social complexities generated by the use of shared services across contexts. The term social paradata has recently been introduced to cover the referencing of data generated by user interactions with digital objects. It was originally adopted by the National Science Digital Library but has been used increasingly in the context of education. In the context of our work on the iTEC Widget Store it has been used to cover, not only user interactions with widgets, but also their expression of value in terms of ratings, reviews and tagging.

The main purpose of encouraging these social annotations is that they can help the community to engage more fully in the utilization of the widgets and their development. Widgets should also be catalogued in a way that makes them findable and usable in terms of their use by a specific community united by geography, social or work identity, or educational interest. Social paradata

---

\[^2\] http://nsdl.org
can be used for this purpose in search, discovery and retrieval, and can also be used to deduce metrics such as popularity or similarity to other widgets.

2.6 Describing widget functionalities

Complete reliance on social paradata is not viable, not least because, particularly at the outset, the richness of available data may be insufficient. It is therefore necessary to “prime the pump” with other descriptions. To this end iTEC WP10 has developed taxonomies that describe the functionalities of tools. The taxonomy developed for functionalities is not specific to widgets, but rather describes the functions of tools in a general way in order to maintain maximum flexibility. The critical benefit is that it allows tools (in this case widgets) to be mapped to the functional requirements of learning scenarios (WP2) and learning activities (WP3) as the same taxonomy is in use in those areas of project work. It has also enabled the store to used as one of the services harvested by the SDE or recommender service being developed under work package 10.

The concept of weighting which was proposed in D8.1 in terms of affordances has been transformed into the present focus on functionalities, in accordance with the requirements of WP10. The taxonomy originally used for affordances is similar to that of the functionalities so conversion at the conceptual and technical level was simple to accomplish and can be used in the same way. The new set of functionalities can be seen in appendix n. A mechanism for defining the functionalities of widgets has been developed and integrated into the iTEC Widget Store. The store service and user interface have been extensively developed through the year ready for Milestone MS27 (Community Widget Server Released). Direct web access to the store is available at http://wookie.eun.org/StoreClient.

2.7 User stories for the iTEC Widget Store

The following simple user stories have been authored by WP8 to represent the way in which the iTEC Widget Store can be used, providing a step by step description of the processes established by the store in order to achieve a particular goal. For the purposes of these stories there is only one type of user who is already registered and signed in. It is assumed that the user is accessing the store through their shell and that sign-in has been carried out automatically.

Story 1: Searching For Widgets

Upon first entering the store the user is presented with the following screen (Figure 3)
The user wishes to find a chat widget to do this the word “chat” is typed into the search box at the top right and the following screen is returned displaying any widgets in which the word chat is found in the search indexes (Figure 4)
The user could have used the predefined searches at the left of the screen. These predefined searches are described as either categories or tags. The categories are based around paradata (as already defined in this document.

The tags are a list of the most popular user generated tags attached to particular widgets.

**Story 2: Viewing and Tagging**

The user has now found a widget of interest and wishes to know more. Clicking on the widget brings up the different screen showing the tags, ratings and reviews for a widget. In this case the user has selected the Youtube Widget (Figure 5).

The widget is tagged with some user-generated tags, and rated. There are also user reviews for the widget. The functionalities of the widget have already been defined and are visible here.

![Image of the Widget Store Viewing and Tagging Widgets](image)

*Figure 5: Widget Store Viewing and Tagging Widgets*

First however the user wishes to see the actual widget in action. Clicking the view button overlays the widget as shown in the following screen shot (Figure 6).
This widget is fully operational and displayed under a special context key (*sharedDataKey*) meaning that activity here is treated as demonstration data and will not interfere with any activity that people are using the widget for in the actual shell which the user is currently logged into.

If the user is happy with the widget then they can if they wish get the embed code for the widget to include it in a page manually, as is shown in the following screenshot (Figure 7)
The user can also tag the widget either by selecting from the list of tags that users have already generated or by creating a new tag. The existing tag is done by selecting from a dialog, as shown in this screen shot (Figure 8).

---

**Figure 7: Widget Embed Code**

**Figure 8: Widget Tagging**
Story 3: Uploading

The user has a widget that they wish to upload to the store. They navigate to the front page and click “Upload New Widget”. This takes them to the following screen (Figure 9)

The user clicks the browse button. They can then select their widget file.

It should be noted that the process for including open social gadgets and imports is slightly different and is described in the widget development strategy part of this document.

When they have selected their widget file upload begins immediately. If the file is large then the upload is tracked and the percentage of upload is shown underneath the browse box.

As soon as the widget upload is complete the user is automatically redirected to the functionalities section of the store. The user then chooses from six lists of functionalities as prescribed by the iTEC taxonomy for tools.

When a functionality is selected, a box representing it appears and the user then moves this box to indicate how relevant the functionality is as is shown in the following screen shot (Figure 10).
The user clicks done and is then taken to the widget view page where the widget can be tagged as appropriate.

2.8 The Widget Store service based upon Edukapp

The iTEC Widget Store as seen by the user is actually a client which accesses a service which manages the data for tags, functionalities, reviews and ratings. This service is based upon an open-source web application called Edukapp\(^3\), initially funded by the Joint Information Systems Committee (JISC) in the UK, but which is being developed as a collaboration with iTEC (through the University of Bolton) and the European Commission funded ROLE project (through the Knowledge Media Institute of the Open University UK). This offers all projects the advantages of pooling resources towards a common goal, and of enhancing the prospects for sustainability of project outcomes.

The server exposes a set of calls that can be made remotely by a software client in order to perform the following actions.

---

\(^3\) See http://widgets.open.ac.uk:8080/
• Search for widgets
• Get individual widget information (extended profile including all reviews, tags, functionalities and ratings averages)
• Get user information
• User sign-in
• User registration
• Widget upload
• Media upload with automatic widget creation
• Tagging widgets
• Adding reviews to a widget
• Assigning functionalities to a widget
• Adding or updating a user rating for a widget

As well as contributing to the core capabilities of Edukapp, WP8 has also developed a number of iTEC-specific extensions to its capability in order to meet the requirements of the project. In order for the iTEC Widget Store to be fully independent of Wookie, WP8 separated the Edukapp kernel so that it communicated with Wookie solely through the REST API. It was also necessary to develop a means of representing and setting functionalities of widgets, as well as introducing date management capabilities. Some extensions to the data model were also required to address iTEC specific meta-data requirements.

2.9 Extending the store API to describe widgets with social metadata and paradata

The present release of the iTEC widget infrastructure extends Apache Wookie in a number of ways in order to support the functionality required by the project. A store API has been established as a separate service which is located along side Wookie itself. This extends Wookie's capabilities with meta-data for each widget beyond the meta-data associated directly with the widget in its config file. As a result the store is able to provide the following extensions:

**Ratings for Widgets:** This enables each user to rate a widget. Each user has one rating record per widget which can be updated, and the ratings of all users can be aggregated (averaged)

**Reviews for Widgets:** Reviews are composed of a block of text which is associated with a user record and a widget record. The time of creation is recorded.

**Tagging:** Tags can be created by users, and those tags which have already been created can be re-used by other users.

**Functionalities:** These were referred to as affordances in D8.1. They conform to the taxonomy for functionalities developed in year two in wp10 and wp2. They can accept a weighting value.
Inter store sharing of social paradata via SPAWS

Several Widget Stores are currently being developed tailored towards education. These include a UK HE Widget store at the Open University, the ROLE Widget Store managed by IMC, and the ITEC Widget Store. While each of these stores is being developed using common standards and common code from Edukapp, they are operated independently by different organisations, and customized for a particular audience. One limitation of this approach is that paradata – reviews by users, ratings and ‘likes’, and aggregate download statistics – is collected separately by each store, even if this refers to the same widget. Sharing this paradata between stores would add value for users of all the stores. To meet this need iTEC has adopted the SPAWS library for sharing paradata, developed in parallel with iTEC by the University of Bolton.

SPAWS uses the Learning Registry network for publishing and retrieving paradata. The Learning Registry is a US Department of Education-backed initiative to create a network of distributed metadata sharing “nodes” - each based on CouchDB - that can distribute records globally. SPAWS can connect to any of these nodes, which can then further distribute the paradata records across the network. SPAWS provides a set of standard paradata classes for user reviews, statistics (downloads, embeds, views, likes) and ratings, and can be extended with other paradata models. The ITEC store periodically publishes paradata using SPAWS, and also periodically retrieves and caches paradata from the network using SPAWS. This means that, for any given widget in the ITEC store, there can be both internal paradata, and external paradata (typically from ROLE and the UK HE Widget Store, but potentially others). For ITEC this provides several advantages:

- It increases the amount of paradata available for the widgets in the store, helping to “bootstrap” the store
- The statistics paradata can be used to improve the ranking of widgets in search results and browse views
- Widget developers can obtain information about their widgets, including popularity and user feedback, from all of the stores they are published in

This provides enhanced information to users for a number of purposes, which may be represented by the following brief user stories:

“As a widget store user, when I browse a widget store, I want to be able to read reviews by other users, irrespective of which store they downloaded it from”

“As a widget store owner, I want to be able to rank widget popularity based on downloads from other sites”

“As a widget developer, I want all the reviews and download stats to be available to users, irrespective of which widget store they are using”
For ITEC the main development effort needed was to integrate SPAWS into the ITEC store. This was relatively straightforward, and required only that existing classes representing items such as user reviews were harmonized with the SPAWS data models. SPAWS has been connected into the main store controllers, and is configured using the ITEC store configuration file:

```bash
# SPAWS support
spaws.enabled=true
spaws.node.location=http://alpha.mimas.ac.uk
spaws.node.username=fred
spaws.node.password=flintstone
spaws.submitter.name=edukapp
# Cache duration in millis for external data retrieved
# using SPAWS: default is one hour, or you can set it
# to 0 for debugging purposes
spaws.cache=3525000
# The interval in days between publishing widget stats using SPAWS
# the default is 1 (daily)
spaws.stats.interval=1
```

## 2.10 Interfaces with other iTEC services

The outputs of WP8 described in this deliverable form part of a wider iTEC infrastructure, and work has been carried out to ensure that WP8 contributions are satisfactorily integrated into the wider system. An iTEC cloud of services has been established, all of which are aware of each other. A single user identity has been established for all these services, enabling a user who is logged on to a .LRN or Moodle shell to transparently access the range of services provided by iTEC.

The iTEC Cloud is made up of the following four components, all of which are integrated with the iTEC widget store:

- User management and Access Control (UMAC)
- The People and Events Directory
- The Composer
- The SDE

The present release of the iTEC Widget Store has been integrated with the iTEC cloud in a number of ways. Firstly it integrates the the UMAC (User Management and Access Control) to provide transparent access to the iTEC Widget Store without further authentication if the user is already authenticated on an iTEC shell. Secondly widget descriptions are integrated with functionalities
and paradata harvested by the SDE (Recommender Service). Thirdly, the iTEC Widget Store API is made use of by the iTEC Composer to create, tag, retrieve widgets. Integration between the iTEC Widget Store and iTEC Shells is achieved by means of the Client Widget which enables the store to be embedded in the target platform.
2.11 Preparation for pilots

Technology is one thing. Real teaching practice is quite another. WorkPackage 8 has worked closely with other aspects of the project, including scenario generation, but also working with real teachers to get an insight into the particulars of practice, and how widget technology might be integrated to address the organizational challenges of classroom practice. On school visits, for example, analyzing stories of how teachers sometimes confidently embrace technologies, but develop quick workarounds in case things don’t work have been very valuable in thinking flexibly about the variety of ways that ITEC Widgets and the store might be used.

From these investigations, it is clear that many teachers have favourite tools and techniques for simple activities conducted from the electronic whiteboard: for example, the use of simple games from resources like Classtools.com. It is also clear that the capacity to share different techniques is restricted by the capacity to curate favoured toolsets and distribute them. The curation facilities of the Widget Store would appear to propose a solution to this personal organization problem for teachers. Other instances of such data gathering from teachers include the need for easily grabbing data from personal devices from students (particularly photographs). Here problems of interoperability can present real challenges for classroom and activity organization.

Through insights like this, the focus has gradually moved toward preparing the store functionality ready for trials with teachers, includes interfaces for the functionality described in these kind of user stories. In accordance with the widget development strategy described above, additional tools are being developed to allow greater flexibility for users to generate their own widgets, while project staff will also develop some widgets to prime the store with widgets relevant to scenarios.

Striking the balance between technical development as an enabler of new practice and new thinking and technical development as a necessary support for scenario implementation has meant that large-scale piloting with teachers, beyond small-scale discussions and investigations, has only become possible quite recently. Only when tools are sufficiently mature can large-scale pilots be conducted. A first step toward establishing a large-scale community of teachers to develop widgets and their infrastructure was taken in the first teacher training session, carried out in September 2012.

These sessions provided important indications for the direction of future technical work. Firstly, the skills of embedding content into electronic learning contexts has become fairly well-understood and widespread, through the use of YouTube videos within learning platforms and the embedding of web content within Interactive Whiteboards. Teachers are becoming familiar with innovative apps and resources for embeddable tools like widgetbox. Once again, as with the initial investigation of teacher practice, there is currently no way of curating favourite tools or resources. When shown ways of embedding favoured tools within the curation environment of the ITEC Widget Store, it was clear that there was considerable enthusiasm. Furthermore, the complexities and scale of engagement that the widget store would have to deal with became apparent. User channels, favouriting as well as commenting mechanisms built into the store now appear as a central element the infrastructure for making the store usable on a large scale.
Given that much early design work on ITEC and widgets had focused on provisioning widgets centrally by the project (this in response to the perceived lack of widgets in the early stages), this sudden explosion of tools on the Widget Store was an important sign that a community-driven, individually-focused approach to curating widget-based resources is likely to be the most effective way in which the widget store can support the communities of teachers.

Having said this, the appeal of curation is one of personal control over technologies that work for individual teachers. When teachers are asked about tools they would like to see, the responses are encouraging for their innovativeness. “Robot controllers”, “Speech Synthesizers”, “Artificial agents”, etc. It may be that given increased control over their technologies, the pedagogical imagination of those who do the teaching is inspired to try things that they themselves can take ownership of.

This last point is significant, for it highlights an important stage in the journey of the project. From investigating an expert-driven programme of widget population, we have arrived at a community-driven facilitation for widget population. This is not to say that there aren’t some tools and functionalities which need to be contributed by the project team (particularly those tools and functionalities necessary for the running of scenarios). But it is to indicate a creative tension between expert-driven pedagogical and technical design and teacher community-driven innovation.
3. INNOVATION AND ENRICHMENT OF THE MASHUP SHELL

ITEC, in aiming for innovation in the classroom, has primarily focused on the creation of innovative pedagogical scenarios. The nature of innovation itself has been a focus within the 2nd year of the project, and Work Package 8 has taken a lead role in helping to shape ITEC’s broad definition of innovation as:

“the process of responding to educational challenges by designing solutions that benefit stakeholders”

Which leads to a more specific definition for ITEC:

“potentially scalable learning activities that provide beneficial pedagogical and technological responses to educational challenges and opportunities.”

The key challenges for Work Package 8 in this definition lies in the central notion of process. Widgets are not processes but products. They may be considered as part of a designed solution providing benefits to stakeholders, however, widgets are products which can enable other processes, including the support of existing scenarios, or in the generation of new scenarios. Therefore, a balance must be struck between the implementation of products (in the form of new widgets) and the support of those processes wherein the innovation that lies at heart of ITEC resides.

In order to understand the process of innovation, insight is required into the challenges of teaching with technologies. It is for this reason that innovative widgets which have been designed have focused on the raising awareness of what is newly possible rather than what may be gained elsewhere. In particular, this has included the investigation of ways of integrating real-time control and hardware devices such as (for example, robots and controllers) into the widget infrastructure. This work has been done in the light of the fact that classrooms are places of shared experience. Most widgets, however, provide individual experiences (although some, like chat, afford ways of sending text messages or conducting activities in real time). With learners in a shared context like a classroom (or indeed any shared space) there is an opportunity to explore the balance between personal control by learners and coordination of shared social experiences by teachers.

This work has been experimental, and trialed largely within the context of the project meetings and demonstrations, but its purpose has been to drive innovative vision of what might be possible beyond the norms of experience of personal technology as it stands. In this way, the classroom of the future, as it is envisaged by ITEC can provide valuable insight into the deficiencies of current personal technologies and the ways in which new technologies might meet the needs of communities seeking technologically-enhanced shared experiences.

Beyond these investigations into what might be possible with specific technologies, the priority of ITEC has been to create a platform for nurturing teacher-led innovation. The Widget Store’s functionality in supporting personal curation of tools allows for teachers to develop their own ideas and inspire new developments.
3.1 Integration of Physical Devices with the Widget Architecture

The rationale for moving beyond the Web browser

The principal motivation for work on physical devices and new kinds of technologically-empowered shared activity in iTEC is to address the question: “if we are all together in the classroom, why would we choose to look at individual screens, rather than each other?”. One of the research challenges in the project has been to explore ways in which activities with personalised devices might be effectively coordinated by teachers. When widgets are displayed on individual screens (for example, in the VLE), each learner potentially can have a different experience, and this leads to problems in the coordination of pedagogic, effective differentiation of activities, and sensitivity and responsiveness to individual student needs. These translate into classroom management challenges. We propose that the appropriate response is to recognise that the learning dynamic of the classroom is determined by a convivial human situation where the principal driving forces are human attachments between friends, enemies, and with the teacher (who of course can be either to different pupils!). Work has been done on studying the ways that physical devices in the classroom can be used to strengthen attachments between learners and deepen the meaningfulness of their experiences. These are often not screens but rather ‘controllers’ for rich shared activities.

Overcoming the disconnect between human attachments in the classroom and individual experience of the technology has led to a technical solution which addresses both the advantages of personalised technologies and the challenges presented by a deeper understanding of attachments. The fundamental component of this technical design is the increasing real-timeness of web-based communication.

The 'real time Web' as an enabling technology

With the emergence of socket-based standards which allow for ultra-fast and timely communications between services and tools, real-time web-based communications are fast becoming a reality. This new functionality, which is one of the core functionalities behind what might be referred to as the ‘real-time web’, with the W3C WebSocket standard being built-into the HTML5 specification, new technological affordances can be made available to teachers which may allow them to address some of the challenges of attachment and control whilst integrating the online environment within classroom environments. WebSockets allow for the establishment of connection-oriented data transfers between web-based applications providing significant speed improvements over existing techniques which employ polling for new data. In effect, the efficiency of a socket-based communication approaches that of the connection-oriented TCP/IP communications upon which the web is built. Supporting the WebSocket protocol, open source
technologies like the V8 JavaScript compiler has been developed into the NodeJS tool as a way of providing highly efficient web servers written in JavaScript. A number of libraries, such as the Socket.io (http://socket.io) library, running on the NodeJS platform (as well as other web platforms such as PHP) have added WebSocket functionality.

**A pilot real-time web widget implementation**

Figure 11 shows a configuration of technologies which has been developed for iTEC. The technical configuration comprises a Wookie Widget Server (A) which provisions tools to learners and teachers, thus addressing the need for tools to be ‘ready-at-hand’ in the same way as traditional classroom resources. However, these tools are configured as HTML5 ‘apps’ which are able to send Socket-based communications to one another, and where those socket-based communications are coordinated by a WebSocket server (the server used in the pilot runs using the JavaScript engine). The timeliness of the socket-based communications means that the NodeJS services acts as a ‘satellite’ (B) which bounces communications between the different users of the widget. The real ‘service’ that the learners connect to and interact with is provided by the teacher (D) in the classroom, who operates a richer tool (E) than the learners have access to, but which responds to the learners’ interactions. Data transferred between users and the NodeJS server are very short control messages. This means that total data traffic is relatively low, even with large groups: an individual node service is rarely handling more than 50 simultaneous messages, meaning latency is kept to a minimum. In pilot testing with groups of participants ranging in number from 30 people,
to small groups of 5 or 6, this ‘rich tool’ served to control the making of music and the movements of a robotic device in the classroom. Participants in the activity use their smartphones, tablets or PCs as ‘controllers’ instantiated either through the Wookie Server or simply through a web page, where messages from the controllers are relayed over the web instantaneously (using WebSockets) to a central PC in the classroom which in turn controls the production of music, or the coordination of the robot.

3.2 Import features of the store for easy creation of widgets

The curation functionality of the Widget Store goes beyond the assemblage of existing tools. Individual users can add new tools easily through new interfaces which support the wrapping-up of web pages, Flash files, or embeddable components (i.e. other widgets from other providers). An example of this is shown below.

A user browses to the online repository of http://widgetbox.com. Finding a calculator widget they wish to curate within the widget store, they select and copy the embed code. Within the Widget store’s interface, they click on the ‘embed’ tab, and are able to paste in the embed code, adding data about the widget’s description and size. Upon completing this process, the user’s widget is made available to all other users of the store, whilst also featuring in the individual user’s personal curated list of widgets (Figure 12)

Figure 12: Widget Selection and Embedding

The Widget concept has rapidly developed online as users have become increasingly familiar with mashing-up and embedding content in platforms ranging from YouTube and Google to blogs and wikis. There is an increasing range of resources for facilitating the authoring of widgets. In essence, such resources and tools are out-growths of web authoring tools, since widgets are basically web-pages. Some of these tools have been developed in collaboration with the technical work on iTec,
including the WidGAT widget creation tools and the MyCocktail widget authoring tool from the EU Omelette project (Figure 13).

![MyCocktail Widget development Environment](image)

There are different kinds of widget tools that can be considered within the Wookie architecture. They are:

- **Single user tools without data persistence** – for example, calculators, timers, etc. These are tools which may simply be grabbed as embeddable objects from the web and which afford simple functionalities. They provide a simple function and don’t store their data. Such tools can be embedded and do not need to be instantiated within a shell.

- **Single user tools with data persistence** – for example, sticky notes, to-do lists, etc. Within Wookie, these tools exploit Wookie’s ‘shared data’ which allows them to store their state relating to an individual’s instance of the widget. Such tools will only work if instantiated within a shell context.

- **Multi-user collaborative tools** – shared editors, chat

Within the Wookie environment ‘proper’ (i.e. widgets designed to run using Wookie features), these tools are similar in nature to single user tools in that they use ‘Shared data’. However, external web tools (for example, Etherpad), can also provide similar functionality maintaining state through an external service. The ability to wrap such tools into W3C widgets creates a very simple way of creating multi-user tools without having to get to grips with the intricacies of Wookie’s shared data.

- **Single user Front-end tools to large-scale multi-user systems**

Simple widget tools for accessing large-scale services (for example, calendaring or workflow management) may be wrapped up as W3C widgets and placed in the store. The ability to embed
these tools in a variety of contexts can allow users to customise their working environment bringing the tools they use ready-to-hand.

### 3.3 Widget creation tools assessed

Bearing in mind these categorizations of widget tool, ITEC has been closely involved in the development of a number of projects for both curating and developing widgets. JISC have supported widget creation with WidGAT tools as part of a strategy feeding into their support for the establishment of a UK-based educational widget store, EduKapp. Within the EU, ITEC work has been closely related to the design and implementation of the MyCocktail tools.

MyCocktail includes tools for creating widgets which aggregate social software services from Google, Twitter, YouTube and many others. It features an easy-to-use interface that will generate a W3C widget file that can be uploaded to the ITEC Widget store. MyCocktail can produce either single-user or multi-user widgets (Figure 14).

![MyCocktail Widget Download](image)

WidGAT has been specifically designed for creating accessible tools. It provides a template-driven approach which can generate either single user tools without data persistence (for example, a timer), or multi-user tools with data persistence (for example, to-do lists).

Commercial developments too have moved the widget agenda forwards. These include the adoption of the W3C widget specification by Opera and the emergence of commercial widget services affording embeddable functionality across a range of contexts (for example, [http://widgetbox.com](http://widgetbox.com)). Commercial widget services provide rich repositories of educational widgets which tend to have attractive interfaces and often innovative functionality. However, whilst some of Opera’s widgets are W3C, they tend to exploit specific features of the Opera
platform which make their incorporation into the ITEC store difficult. Far more effective are the embeddable widgets provided by services like WidgetBox which can simply be integrated within the Store. Both kinds of widget fit into the single-user without data persistence category. WidgetBox also provides a well-designed widget production toolkit.

The fact that the W3C widgets of Opera are less easily imported into the ITEC store than the embeddable but non-W3C widgets of WidgetBox highlights an important point. Widgets are web pages. The technology of the web affords many mechanisms whereby HTML code can be injected into a particular context. The use of iFrames which point to some external hosted service is in many respects the quickest and most convenient way of producing the HTML within its context. Whilst it may appear desirable that the HTML for a widget is actually contained within the widget package itself (as it is in Opera Widgets), and indeed in terms of manageability this may be most desirable, the flexibility of the web allows for many different approaches for getting content delivered by the Wookie server.

3.4 Rationale behind widget development

Initially the project did not have a development strategy for the development of new widgets, but as time has gone by it has become increasingly apparent that a strategy is required, for two reasons.

Firstly, when the project plan was written it was expected that there would be an abundance of good quality open-source widgets from which project staff or the teachers, could select and easily importable into the store. Whilst it is true that there are widgets available now, the process of importing them is less than perfect. This is primarily because few people have fully adopted the W3C standard for widgets. Many have adopted it partially and many have allowed for importing of W3C widgets into their own system, but this partial adoption of the standard has caused difficulties for the project.

Secondly, there is a need for good quality widgets, which are appropriate and directly useful for the project’s scenarios and learning stories. This presents the opportunity to create some good quality, applicable, tools which support the learning we are promoting, which are easy to use and are badged as iTEC widgets. These widgets will fully support the W3C standard so will, in principal, be usable outside of the context of iTEC as well.

We also plan to support user generated widgets as part of the development strategy. Teachers within the project will be trained and encouraged to build widgets from other media types, such as flash and java applets, making use of the tools described in the store section of this document, which makes the process of transforming a Web resource into a widget simple, by using an import feature in the store.

3.5 The iTEC widget development strategy
The purpose of developing a widget strategy for the iTEC project is to use the best means possible to populate the iTEC Store with as many relevant widgets as is feasible with the given timescale. It will guide support for widget creation in the coming period of WP8 activity.

The widget strategy has three strands:

- Develop widgets within the project that are useful to the iTEC pilots but are generic enough to enable them to be re-used for different learning stories.
- Finish and deploy the store upload and import features that will allow teachers to create widgets from existing content or tools.
- Find existing widgets that can be imported into the store.

These strands of widget development are planned to progress over the coming year and the outcomes of them will be documented in D8.3. The details of the strategy are provisional in the sense that it will respond to the evolving requirements of iTEC users and pilots, and the services and technical opportunities which become available during the remainder of the project.

**Widget Development**

The project will develop a set of widgets in W3C format, which make use of User Management and Access Control features to make them seamlessly accessible to iTEC users. The tables provided in Appendices I, II and III of this document outline the proposed W3C widgets, which will be installed, delivered and managed by the wookie server through the iTEC widget store (http://wookie.eun.org/StoreClient). The selection of the widget types to be developed was based on an analysis of the iTEC scenarios. As the project has run, the Widget Store has provided a way in which rapid development of tool collections within a learning platform can be assembled. For example, at a workshop with members of other workpackages, a particular learning scenario was chosen. Within 10 minutes, the scenario could be ‘instantiated’ as a resource within a learning platform (Moodle) with tools and instructions as to how the scenario might run and how teachers might use the tools provided. Missing tools, or tools which only partially met the requirements of the scenario could quickly be identified. Indeed, owing to the functionality of the Widget Store, some ‘missing’ tools could quickly be uploaded and instantiated. However, this kind of exercise highlights the value of the approach taken, since with an easy and rapid way of collecting assemblages of tools within a learning platform, group reflection is the most powerful aid to widget development and refinement, as people look at it and ask “could this work?”, “what might be better than doing that?”, “how could the children be organized to use this widget?”, “would this activity be done in home or for homework?”, etc. Through this kind of process, a variety of widget improvements have taken place.

This led to a realisation that generic/general purpose widget tools were the most important.

**Model for Developing a Community of Teacher Creators**

The intention behind opening up the creation of new widgets through the store, is to encourage a community of sharing of content and tools between the users of the store (that is primarily teachers within iTEC). Teachers to be able to upload content in a variety of formats, tag it with
appropriate information and for the system to turn this content into a W3C widget. This widget will then be available through the store for others to use, rate and review. For example, teachers will be able to upload mathematical models they create in GeoGebra, or to include any Flash applets that they might have found on the web or developed themselves. Licensing issues will mean that a review process for content will be required.

The store will have upload and import features for a variety of formats. The file formats supported will be:

- Flash files (.swf files)
- Java Applets (.jar files)
- HTML files – single file only for import
- Image files – web formats such as Gif, JPEG, PNG
- Movie files – web formats such as QuickTime and MPEG
- Sound files – mp3, air, QuickTime, MIDI etc...

Imports will be presented to the user as standard file uploads in the same way as new widgets are uploaded. The upload script will detect the media type for the file and then create a widget appropriately. After the initial upload additional information for the widget will be required from the user. This will include a name for the widget, the size of it, a description and any licence information. Finally the user is given the opportunity to tag the new widget with any functionalities that are appropriate and indicate how relevant they are on a scale of one to ten.

The administrator of the store will then be notified of new widget tools or content to be reviewed. When the review is completed, this new widget will be available in the store for all users to rate, review and tag. It will also be available to shells through the wookie server.

The production of widgets by project partners is helpful in “pump priming” the widget development strategy, but it does not in itself constitute a long term, sustainable mechanism for providing a large scale of useful widgets. In year 3 of the project WP4 intends to introduce teacher training activity with widget development tools to selected “Ambassador” teachers. These will be asked to make at least one widget themselves, and encourage other teachers to do so as part of their work to facilitate the growth of a community of practice. A hundred teachers are expected to participate in this training in the lifetime of the project. This will yield 100 simple, yet potentially valuable widgets. If each teacher encourages at least one colleague to use the tool, this will double the numbers, and while some teachers may not be active contributors long term, it may be expected that others will contribute significantly. A more speculative possibility to be explored is the establishment of a collaboration with an existing community of developers of similar applications, such as Android Apps, who could take a leading role in larger scale widget development.

### 3.6 Widgets providing iTEC cloud interfaces

Tools to access and user the iTEC services are also being created as W3C widgets. In all cases the tools with a user interface are being developed as widgets to allow them to be easily integrated
into the shell. Indeed the iTEC Widget Store itself is being developed as a widget, although the user is not aware of this. As far as the user is concerned they navigate to a page in their shell in which contains the store. They can also find the other services, such as people and events directory on a page in their shell.

The rationale for doing this is simple. Whilst, it would have been easy enough to create Moodle pages, or DotLRN addons that directly accesses the services, each time a new shell is introduces a new plugin or connector would need to be created with specific integration features to allow iTEC services to inhabit the shell. By developing the service user interfaces as widgets they are far more easily transported to new environments. The access method for each service is seamlessly the same. Even in Moodle, since the connector is already in place, each service simply has to be included via the standard widget inclusion method (As described in D8.1).

The following are the type of widgets that are being developed as central iTEC Cloud Interfaces.

- Administrative tools for user registration and login
- A widget interface to the People and Events Directory
- A search widget for EUN’s LRE (Learning Resource Exchange)
- A widget interface to the iTEC Composer
- A widget interface to the iTEC Store

Finally other prototype widgets are being created by work package three. These are designed to be specifically with their learning stories and are documented with the work package three deliverable.
4. MAINTENANCE AND SERVER MANAGEMENT

There are a number of potential risks in running a large-scale educational widget store for European schools. These include:

- management of large numbers of users
- management of large amounts of content
- management of data exchanges through the Wookie server
- management of metadata and paradata
- management of associated services to support specific widget functionalities (i.e. specific project widgets)
- management of deployments of updates to the software

Given that widget-based learning activities are effectively ‘mission critical’ for teachers using those widgets within lessons, WorkPackage 8 has been actively involved in processes of:

- Structuring the technical architecture so that key areas of risk are separated and can be managed individually. In the first instance by separating the Wookie server from the Widget Store.
- Ensuring that participation in contributing and using the widget store takes place within the context of the ‘walled-garden’ of the ITEC cloud
- Introducing software changes that would facilitate decentralising data management as far as possible whilst maintaining the benefits of community engagements (for example, the use of SPAWS)
- Providing a rich ecology of tools and personal ownership so as to minimise emphasis on centrally managed tools.

Currently, the iTEC Widget Store has been established as an active server on the iTEC cloud. The Widget Store service is comprised of the following components.

- Wookie
- Edukapp
- Apache Solr
- Apache Shindig
- Store Client

These services are hosted by EUN, and the performance of the services will be evaluated by a six monthly review drawing on data ranging from user evaluation to technical server log files.

An active process of understanding the deployment challenges has been engaged with in the first six months of operation of the store. This has highlighted some challenges which are being addressed. For example, initial deployment of the iTEC Widget Store and related software revealed some upgrade difficulties. Version 0.8 was being run by EUN, with the iTEC Teamup widget was running on that server. This makes use of Wookie’s shared data feature to store it’s
data, but when the Wookie server was updated this data was lost. It was possible to roll back to the previous installation, and subsequently to migrate Teamup to the new server. Nevertheless, an important issue was exposed and technical solutions and process were put in place to make sure that similar problems are not repeated. Update scripts are now created for the databases of Wookie and the Widget Store for each new release, allowing the server to be updated directly. An upgrade strategy has also been established so that database upgrades can be carried out transparently for users, and so that server downtime is minimised. Processes have also been put in place to ensure that widgets and their associated data are not lost when the server is upgraded.

Such processes will form an important part of the emerging strategy for a sustainable large-scale service. It is important that optimal solutions both for providing shells and for serving widgets to schools are investigated. Currently, the efficacy of providing a central shell, Moodle, which works together with the centrally-hosted Wookie server and Store, all hosted by the EUN is being explored. Whilst this approach is likely to have its drawbacks, it can nevertheless highlight the technical priorities for more optimal provisioning whilst also providing a public demonstrator of the ITEC technologies. Indeed, some national coordinators are experimenting with different kinds of configuration, including local shells (still Moodle, with the Wookie plugin) working with a central store. Other permutations and combinations are possible and likely to be explored.

The management of centrally-served individual widgets can present difficulties in the short term. These widgets are important in the early stages of the project to inspire teachers into new pedagogical practices. However, as a teacher-driven personalized curation of tools becomes more mature, centralized widgets are likely to become less of a priority. Having said that, the Widget Store can provide a powerful mechanism for the dissemination of innovative tools which may be served from a central point in the first instance. In the long term, there is no reason why such tools should be managed by the project, but instead could be managed by individual teachers or ministries with the technical infrastructure and skill to do it.
IMPACT ON PRE-STANDARDISATION

The work carried out by WP8 makes use of interoperability specifications in all its aspects. In particular, The Apache Wookie Widget Server is a reference implementation of the W3C widget specification, and the Widget Store provides a showcase for services created using the specification. The project has also integrated IMS LTI (Learning and Tools interoperability) into Apache Wookie.

The challenge facing W3C widgets is no longer a technical one of how to formulate the specification, but rather one of adoption levels, and problems and opportunities we may have with that. The consequence is that our preference is comply with the W3C specification as it stands wherever possible, partly to promote interoperability, but also because as to depart from the specification threatens the reliable functioning of the iTEC project infrastructure.

Consequently it is not part of its purpose to generate new standardisation initiatives, but rather to demonstrate the effectiveness of an interoperability strategy, and to create reference implementations of interoperability specifications. In the course of this work WP8 maintains close contact with W3C, the authors of the widget specification used by the project. No additional requirements have yet been identified in the course of project work. As and when this occurs they will be fed into W3C and IMS. The consequence is that this task (T.8) has so far represented a rather small proportion of project effort.

Having said this, there are a number of issues of relevance to W3C Widget standardisation and interoperability which have emerged through work in the project and which deserve mention. Amongst them, the most significant in terms of its impact on interoperability has been the specification of specific feature sets which restrict the interoperability of widgets. These have been noted particularly with regard to widgets from commercial sources like Opera, that features are declared within the Widget package which can only be addressed by running the widget within the Opera environment. This is clearly a barrier to interoperability.

Other interoperability issues, although not directly concerning the W3C widget specification itself but more a rising trend in the web, concern the increasing restrictions on embeddable components across different contexts. This together with restrictions on the ways in which specific commercial APIs are called (for example, the recent restrictions introduced by Twitter) mean that embedding features from certain popular providers (Twitter, Facebook) presents some technical challenges whose origin is not in the technology but in the business models of the providers of those services.
5. CONCLUSION

The principal achievements of WP8 reported in this deliverable relate strongly to the iTEC Widget Store.

Firstly, the potential of a widget store as the means of enabling individuals and groups to create, curate and deploy collections of services and resources has been clarified. Initial feedback from teachers has shown that there is enthusiasm for widget creation, and that the refinement and optimisation of the mechanisms for curation and selection of widgets will be a major challenge in the coming period. The implications of this the information model have been thought through and have informed the development work carried out.

Secondly, the information model for the widget store has been implemented. A number of different lines of work have contributed to this outcome.

- The store service was created within Apache Wookie, and a client application was created which constitutes the iTEC Widget Store.

- Widget creation tools have been implemented and integrated into the iTEC widget store. These create widgets from: uploaded media (flash and java applets) and from embed codes.

- The Apache Wookie Widget Server used in the iTEC Widget Store has been updated to meet the requirements of the project. As a result of this work the first full graduated release of the software is now imminent.

- Connector framework SDK has been extended to the full REST api for Apache Wookie, making it possible for the iTEC Widget Store to make use of the enhanced capabilities of Wookie.

- A new connector block for Moodle has been created, enabling the iTEC Widget Store to be integrated.

- New widgets have been created, and innovative widget designs explored

The result is a fully functional demonstrator which is ready for trials with the end user group. This provides a solid basis for future pilots and further refinement of the iTEC Widget Store.

The store and related development work is open source, and has been carried out with the strategy of contributing wherever possible to existing open-source projects. Interoperability has been maintained with W3C widgets, and compliance with Learning Registry paradata has been added.
# APPENDIX I: TABLE OF WIDGET DEVELOPMENTS

<table>
<thead>
<tr>
<th>New Widget Title</th>
<th>Description</th>
<th>Ready By</th>
<th>Responsibility</th>
<th>Relevant for</th>
</tr>
</thead>
<tbody>
<tr>
<td>Calculator</td>
<td>Two calculators, one scientific (based upon the opera scientific calculator) and one simple written as a wookie widget</td>
<td>1st July 2012</td>
<td>Bolton</td>
<td></td>
</tr>
<tr>
<td>Modifications and improvements to Current Widgets</td>
<td>Camera Widget – stores files Shared Draw – drawing tools and viewer list Wiki widget – larger and multiple wikis.</td>
<td>25th July 2012</td>
<td>Bolton</td>
<td></td>
</tr>
<tr>
<td>Etherpad</td>
<td>This is a generic document shared editing environment. It already exists as a gadget and will be imported from the ROLE store.</td>
<td>25th July 2012</td>
<td>Bolton</td>
<td></td>
</tr>
<tr>
<td>File Browser Widget</td>
<td>A widget allowing users to browse files which have been created by the camera widget. And in future any widget which allows the saving of files or data to the server. This widget accesses a file manager service which has been developed to simply manage files for the iTEC project.</td>
<td>1st Sept 2012</td>
<td>Bolton</td>
<td></td>
</tr>
<tr>
<td>6 Thinking Hats Widget</td>
<td>This widget supports the “Six Thinking Hats” method by Edward de Bono. It introduces the six different hats (i.e. different views) and stimulating questions. A visual way to show the 6 thinking hats (=colors) with some explanations of their meanings, and the ability for the teacher to select one to be “on” for everybody.</td>
<td>1st Sept 2012</td>
<td>Smart</td>
<td></td>
</tr>
<tr>
<td>Idea Cards</td>
<td>This is a widget that supports a stack of stimulating flash cards. On the front a visual picture inspires students to apply a specific method or strategy (such as interview/observe/measure/collection or brainstorm/brainwriting/mindmapping, ...), on the back the details about the method are provided. The widget can be used on IWB, on student devices and in LMS/PLEs. Could be generalized into a card widget, where you have a stack of cards, with whatever content on them, which can then be flipped through or randomly skipped across.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Random Word Impulse</td>
<td>This is a widget that supports lateral thinking to come up with new ideas. Basically it generates one or more random words that are then used as an impulse for the problem or design task at hand. The widget comes with a list of meaningful words (e.g. nouns that are easy to understand). The shell has to provide further introduction about the different creativity methods (e.g. in an LMS there could be an intro text on a page, on an IWB there would be a .notebook or .flipchart file that introduces the method). One example is the design of a new toaster. If the random impulse generator spits out &quot;beach&quot; students have to think about properties of the beach that could be useful for the original design problem (e.g. at the beach you have to be careful that you are not burnt by the sun -&gt; the new toaster could have sensor that</td>
<td>1st Sept 2012</td>
<td>Smart</td>
<td></td>
</tr>
<tr>
<td><strong>Brainstorming / Share ideas widget</strong></td>
<td>This widget accepts simple words or images and shows them on the SMART Board. It will use SMART Notebook software functionality to display text/images (similar to Twitter or Chat widget). A generic W3C version of the client will be developed</td>
<td>Smart</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## APPENDIX II: PHASE TWO WIDGET DEVELOPMENTS

<table>
<thead>
<tr>
<th>Widget Type</th>
<th>Description</th>
<th>Date</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sticky Notes Widget</td>
<td>A widget allowing users to place their own sticky notes on a page.</td>
<td>1st Nov 2012</td>
<td>Bolton</td>
</tr>
<tr>
<td>Voting Widget</td>
<td>Similar to or perhaps in collaboration with Socrative. However, we already have a minimal voting widget called You Decide as part of the standard installation. The initial development strategy here will be to take that widget and look at adding features, which make use of the new wookie role capability.</td>
<td>1st Nov 2012</td>
<td>Bolton</td>
</tr>
<tr>
<td>Media Handling Widget</td>
<td>This could be a drop box widget. Or it could make use of file storage on the server. File storage can be implemented with third party – open source file saving systems such as jQuery Upload.</td>
<td>Dec 2012</td>
<td>Knowledge Markets</td>
</tr>
<tr>
<td>Leader Board</td>
<td>A leader board which initially will be managed by a teacher, designed for use with in-class games of any sort.</td>
<td>Dec 2012</td>
<td>Bolton</td>
</tr>
<tr>
<td>Forum (draft)</td>
<td>Either using the media server and the people database or linking to a third party forum tool. This will allow cross-shell forums. The ability to link across shells will require this widget to use iTEC's user management and access control system, so effectively this will be a purely iTEC widget.</td>
<td>Dec 2012</td>
<td></td>
</tr>
</tbody>
</table>
### APPENDIX III: PHASE THREE WIDGET DEVELOPMENTS

<table>
<thead>
<tr>
<th>Media Server Widgets</th>
<th>If the media server is developed as part of the iTEC cloud a suite of widgets will be developed to handle file sharing, image sharing and video sharing. This may be one widget, which can handle all, or a separated set of widgets that can be used independently of each other or together. If the media server is not developed then two options are available. Either a third-party open source ajax based file sharing system</th>
</tr>
</thead>
<tbody>
<tr>
<td>April 2012</td>
<td>Knowledge Markets</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Blog Widget</th>
<th>If the media server is available the blog widget will use that as a store for it’s data, linking identity with iTEC’s user management and access system. If however, the media server is not available then an external blog (perhaps wordpress) service will be utilized</th>
</tr>
</thead>
<tbody>
<tr>
<td>April 2012</td>
<td>Bolton</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Generic LTI Widget</th>
<th>This will allow the seamless connection to web-based, externally hosted applications, content or tools that support the LTI standard.</th>
</tr>
</thead>
<tbody>
<tr>
<td>April 2012</td>
<td>Bolton</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Forum</th>
<th>Continuation of development and refinement of the forum tool. This tool will possibly take the most work to complete well. So development will cover two phases. In this phase the final version will be available.</th>
</tr>
</thead>
<tbody>
<tr>
<td>April 2012</td>
<td>Bolton</td>
</tr>
</tbody>
</table>
APPENDIX IV STORE REST API

Authentication

All authentication is done via http basic authentication.

There are three roles: admin, owner and standard user

All registered users are standard users

Standard users can also be owners in the context of a widget they have uploaded or created.

Notes

The prefix for all of the rest calls is /api/rest

The following is a set of tables showing the breakdown of REST calls available from the store. It has been divided into conceptual units which in most cases correspond to a path or resource. This isn’t always the case however, as some paths needed to be extended to allow security to be applied to a particular path. This is particularly the case where a GET call will simply return a list or an object and no security is required whereas a POST call on the same resource would need security.

Widgets

<table>
<thead>
<tr>
<th>Action</th>
<th>Path</th>
<th>Description</th>
<th>Security</th>
<th>Returns</th>
</tr>
</thead>
</table>
| GET    | /widgets/search/{query}/{start}/{rows}/{orderby} | Search 
  {query} is the search term 
  {start} is the row or results to start from 
  {rows} is the number rows to be returned. 
  {orderby} is optional and can be one of the following terms: date, rating or popularity | Open | JSON List of WidgetProfiles |
| DELETE | /widgets/delete/{widgetId} | Delete a widget 
  {widgetId} is the id (integer) | Basic authentication | JSON Message |
<table>
<thead>
<tr>
<th>Action</th>
<th>Path</th>
<th>Description</th>
<th>Security</th>
<th>Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>POST</td>
<td>/uploads/widget</td>
<td>Upload a widget</td>
<td>Basic Auth (owner/admin)</td>
<td>JSON WidgetProfile</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The content type should be set to “multipart/form-data”. The file field should have the name “widgetFile”</td>
<td></td>
<td></td>
</tr>
<tr>
<td>POST</td>
<td>/uploads/file</td>
<td>Upload a flash file and have it widgetized.</td>
<td>Basic Auth (owner/admin)</td>
<td>JSON WidgetProfile</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Content Type: “multipart/form-data” With the following fields:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>- widgetname(string),</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>- widgetdescription(string),</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>- widgetwidth(string),</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>- widgetheight(string),</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>- mediafile(file)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>POST</td>
<td>/uploads/embed</td>
<td>Create a widget file from an embed or some html code.</td>
<td>Basic Auth (owner/admin)</td>
<td>JSON WidgetProfile</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Content Type: “application/x-www-form-encoded”</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Fields:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>- widgetname,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>- widgetdescription,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>- embed,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>- widgetwidth,</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>- widgetheight</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Functionalities**

<table>
<thead>
<tr>
<th>Action</th>
<th>Path</th>
<th>Description</th>
<th>Security</th>
<th>Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>/functionalities/</td>
<td>Level is optional. There are None</td>
<td>JSON list of</td>
<td></td>
</tr>
<tr>
<td>Action</td>
<td>Path</td>
<td>Description</td>
<td>Security</td>
<td>Returns</td>
</tr>
<tr>
<td>--------</td>
<td>------</td>
<td>-------------</td>
<td>----------</td>
<td>---------</td>
</tr>
<tr>
<td>POST</td>
<td>/functionalities/edit</td>
<td>Assign a functionality to a widget with level. Form params: Widget_id, functionality_id, relevance</td>
<td>Basic Auth (owner/admin)</td>
<td>JSON</td>
</tr>
<tr>
<td>DELETE</td>
<td>/functionalities/edit/{widgetId}/{funcId}</td>
<td>Removes a particular functionality from a widget.</td>
<td>Basic Auth</td>
<td>JSON</td>
</tr>
</tbody>
</table>

**Tags**

<table>
<thead>
<tr>
<th>Action</th>
<th>Path</th>
<th>Description</th>
<th>Security</th>
<th>Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>/tags</td>
<td>Get all tags</td>
<td>None</td>
<td>JSON List of Tags</td>
</tr>
<tr>
<td>GET</td>
<td>/tags/popular</td>
<td>Get the most used tags in the list</td>
<td>None</td>
<td>JSON List of Tags</td>
</tr>
<tr>
<td>POST</td>
<td>/tags/edit/{widgetId}</td>
<td>Content-Type: &quot;application/x-www-form-encoded&quot; Fields: tag: string Either creates a new tag or reuses a tag with exactly the same name</td>
<td>Basic Auth</td>
<td>JSON</td>
</tr>
<tr>
<td>DELETE</td>
<td>/tags/edit/{widgetId}/{tagId}</td>
<td>Removes the tag from the widget, this does not delete either tag or the widget.</td>
<td>Basic Auth (owner/admin)</td>
<td>JSON</td>
</tr>
</tbody>
</table>

**Reviews**

<table>
<thead>
<tr>
<th>Action</th>
<th>Path</th>
<th>Description</th>
<th>Security</th>
<th>Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>/reviews/{widgetId}</td>
<td>Gets all of the reviews for a particular widget</td>
<td>None</td>
<td>JSON List of Reviews</td>
</tr>
<tr>
<td>POST</td>
<td>/reviews/edit/{widgetId}</td>
<td>Content-Type: &quot;application/x-www-</td>
<td>Basic Auth</td>
<td>JSON Boolean</td>
</tr>
</tbody>
</table>
Ratings

<table>
<thead>
<tr>
<th>Action</th>
<th>Path</th>
<th>Description</th>
<th>Security</th>
<th>Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>/ratings</td>
<td>Gets a user rating for the widget. The rating is for the currently signed in user.</td>
<td>Basic Auth(Owner)</td>
<td>JSON Userrating</td>
</tr>
<tr>
<td>POST</td>
<td>/ratings/edit/{widgetId}/{rating}</td>
<td>Updates the rating for the currently signed in user.</td>
<td>Basic Auth(Owner)</td>
<td>JSON Boolean</td>
</tr>
</tbody>
</table>

Return Types

All data returned is in the same format – JSON Strings. These are utf-8 encoded human readable strings. The following code samples show’s the construction of the strings for each data type. They have been formatting in the samples below to make them easily readable. The actual data returned by the REST calls contain not tabs or return characters.

Widget Profile

This is actually an extended widget profile which also contains the embed html and the url for a widget instance. The widget profile I contained within.

```json
{
    "widgetProfile" : {
```
"id" : int,
"name" : "String",
"icon" : "URL",
"featured" : boolean(0|1),
"created" : "Date String",
"updated" : "Date String",
"tags" : [ {
    "id" : int,
    "tagtext" : "String"
},
... ]

"activities" : [],
"description" : "String",
"downloads" : int,
"embeds" : int,
"views" : int,
"averageRating" : int,
"totalRatings" : int,
"functionalities" : [ {
    "relevance" : int,
    "id" : int,
    "name" : "String",
    "uri" : "URL",
    "level" : int
},
... ]

"type" : "W3C Widget|Open Social Gadget",
"uri" : "Widget Identifier"
"uploadedBy" : "Name",
"renderInfo" : "HTML to embedd",
"renderUrl" : "URL to the widget instance",
"downloadUrl" : "URL to the widget file"
Message
Message is a just a string returning information about the result of the action.

```
{
    "id": int,
    "message":"String"
}
```

Functionality

```
{
    "id":int,
    "name":"String",
    "uri":"URL Identifier",
    "level":int
}
```

Tag

```
{
    "id":int,
    "tagtext":"String"
}
```

Review

```
"id": int,
"time": "Date String",
"text": "String",
"user": "Username"
}

Userrating
{
    "id": int,
    "rating": int,
    "time": int,
    "user": "Username"
}

All JSON strings return evaluate into Javascript objects.

If jQuery’s ajax is being used to make the call then setting the return type to JSON will automatically convert the strings to javascript objects. If not javascript’s eval function will do the same. For other languages conversion procedures will need to be put in place.