

## Backup/restore Support UNIX → Windows

The client is a reputed organization in storage domain providing efficient backup solutions for Windows platforms. There was a strong need to support backup/restore from/to UNIX based platforms also.

Izel Technologies was chosen for the task given its extensive expertise in the backup/restore domain and more specifically many years of development and testing experience in granular recovery of backup data (file-level as well as image-level).

Business requirement:

- ⤴ Development of an agent that can run on Unix platform and allow to take backups from and restore to the Unix machines.
- ⤴ Integrate the agent's client side with already existing backup software.
- ⤴ End to end testing of both client and agent.
- ⤴ Documentation covering all the usage, functionality description and configuration information.

## Description of competencies & technical skills/expertise demonstrated

### ⤴ Introduction

Backup software is available on Windows platform. It communicates with the agents installed on Windows platform to perform backup/restore. Our developers have an in-depth knowledge of proprietary backup software that runs on Windows and is the centralized management UI for all the machines in network. There is a list of APIs that need to be supported on Unix as per the format used on Windows software.

### ⤴ Challenges

To make Windows management server communicate with Unix, understanding of the proprietary network communication protocol is a must. The development team had spent a lot of time in analyzing the protocol by writing step by step test utilities to validate the understanding. Software on Windows used default wide character size as 2 bytes however on Unix based platforms the default size is always 4 bytes. The development team had spent good amount of time in making the size compatible with seamless implementation.

### ⤴ Technical Skills

The entire development is in C++, Perl and Bash. On Windows, it uses proprietary network and data packaging utilities on Windows. The same data packaging utilities are ported to Unix too in order to use the counterpart calls.

### ⤴ Methodologies

- The requirements for the project were well documented at the time of requirement analysis. The development started once both us and client had an agreement on the expectations of requirements. List of APIs, to be supported, was prepared and reviewed from client. On the basis of that document, the task breakdown and ETA was prepared. The standard Waterfall model was followed with agile touch to it. Iterative model was adopted in the later phase of project. Testing was divided into two major stages. Firstly, Unix side code was implemented and test utility was written to test the standalone code on Unix. Secondly, Unix side code was integrated with Windows side code and second stage of testing was carried out.

## ⤴ **Competencies**

### ◦ **Implementation**

Entire code was written with manageability, scalability and performance in mind. Understanding the limitations of the Unicode compatibilities helped coming up with the best design for performance optimizations. The modularity of the code made it very easy to change any requirement that came from customer later in the implementation. For example on addition of a new API to be supported, one needs to change the code in maximum 3 files. The resource allocation ensures that every module has a shadow developer. This helps in peer reviews and handle situations in absence of one of them.

### ◦ **QA centric approach**

Testing is the most important phase of the project and maximum time is expected to be spend in this phase. Optimizing the code right from beginning is not advisable. This phase is where the project model acquires iteration.

### ◦ **Customer Satisfaction**

Clear, precise and transparent communication – At every stage of the project, the customer was aware of the every proceeding on the project. Daily status updates on portal and weekly calls were part of the project execution. Well documented WIKI was the single page find-the-details for any person interested in knowing any part of the project. Even technical details were captured in a documentation form and provided and updated in timely manner.

### ◦ **Documentation**

The documentation of the agent was completed in two phases - image level backups and file level backups. For GUI part, screen-shots were provided so as to make user understand how it works in the simplest way possible.

### ◦ **Productization**

Apart from developing the product and assuring the quality of it, we also setup a license server, package repository and development involved to achieve the same. At this stage, the software developed gets finally converted to a full-scale product ready to be released into the market.

## ⤴ **Highlights of technologies used**

Microsoft Visual C++, Perl, Bash. The APIs supported were implemented in such a way that the same can be modified easily in case the input/output changes in future. Device driver development on Linux was also involved in the project as a major feature.