

Automating Your Data Collection: Reaching Beyond the Firewall

WHITE PAPER

Automating Your Data Collection: Reaching Beyond the Firewall

“Automating repetitive steps and eliminating those that waste time will increase information worker productivity and save an organization millions of dollars”
— IDC White Paper

Much has been written about the complexity of programmatically extracting data from unstructured sources such as the web but little about the cost of failing to do so. A recent study by IDC revealed that information workers typically spend 17.8 hours per week searching and gathering information at an annual cost of \$26,733.60 per worker.¹

Large companies employing a thousand information workers are hemorrhaging money, as much as \$5.7 million annually, in time wasted reformatting information between applications.

IDC’s conclusion? “Automating repetitive steps and eliminating those that waste time will increase information worker productivity and save an organization millions of dollars.”

Given the shift toward an information economy, the intensity of globalization and hyper-competition, automating your external data collection may be a significant cost savings as well as a competitive advantage.

Fortunately, the technology to automate your data collection already exists. Intelligent agents are software programs capable of navigating complex data structures, extracting targeted data with a high degree of precision, then repurposing the data into an actionable format – exactly the sort of tedious, repetitive tasks that are costing American industry so much money. Solutions range from simple agents with rudimentary intelligence to smart software – highly autonomous agents capable of reading numerous file formats, negotiating security protocols, duplicating session variables, submitting forms, and extracting gigabytes of data from the deepest crevices of the web.

Three Approaches to Data Collection

There are three basic approaches to collecting data beyond the corporate firewall: cut-and-paste, custom programming, and Integrated Development Environments (IDE). Each approach has applications, costs and benefits.

THREE APPROACHES:
Manual Cut & Paste, Custom Programming, or Integrated Development Environments

Data collection suitable for automation is typically repetitive extraction from a known source. An energy trading company provides a good example.

High volume trading of crude oil and refined products, natural gas and power requires monitoring hundreds of information sources – subscription services, internal data repositories, and websites posting production data. Some of these sources may need to be queried only weekly, others almost in real-time. The extracted data must then be structured and imported into a database or data warehouse where it can power other programs and be distributed throughout the enterprise.

In the business of energy trading, a few seconds is a lifetime. Numerous factors affect the price of energy – weather, water flows, and transmission capacities, for example. Data must be continually collected and integrated in order to make split-second decisions.

In this example, the volume of the data collected and the need for near real-time collection and repurposing (translating unstructured or semi-structured data into a structured format) precludes a cut-and-paste approach.

What Makes Software Smart?

Intelligent agents may need to cooperate with other agents to complete their task.

What makes one piece of software smarter than another? There’s still some debate but generally intelligent agents are defined as **Autonomous, Proactive, Responsive, and Sociable**.

Autonomous: Once released, an intelligent agent carries out its mission without further instructions. Most software requires constant prodding from its user – a keystroke or mouse click. Intelligent agents, also known as autonomous agents, exercise greater independence.

¹ Based upon an average annual salary of \$60,000, including benefits

Proactive: Agents are goal-oriented. They are driven by a single imperative: to accomplish their task. They never tire, never sleep, never get surly. They can be thwarted in their mission but never distracted or dissuaded.

Responsive: The more intelligent an agent, the more capably it can recognize and respond to changes in its environment and resolve challenges to the successful completion of its mission. Very bright agents can even learn from their environment.

Sociable: They may need to communicate with other agents to complete their task. They're expected to communicate with their employer what they've learned in the wild. And should a mission fail, an agent needs to debrief on the cause of its failure in order to craft a more perfect mission.

Features of an intelligent agent include:

- Identifying targeted data
- Programmatically navigating the data source
- Reading file formats and contents
- Conservation of external resources
- Self-defense

Identifying Data

There are two methods of identifying data in unstructured sources such as the web: physical and logical pinpointing.

Physical pinpointing requires that an agent's programmer first visit the page and physically highlight the desired data, providing the agent something akin to a geographical reference. It's quick and dirty.

Physical pinpointing has the advantage of being cheap. It's easily programmed using inexpensive tools. Unfortunately, a slight change to the underlying structure of the page – even the reformatting of text – can invalidate physical pinpointing. Ultimately, high maintenance costs may erode the initial savings.

A more resilient method of identifying data is logical pinpointing – a combination of Boolean expressions and pattern matching. Logical pinpointing attempts to orient the agent by identifying data values. It's a technique that accommodates greater changes in the page structure without breaking the agent's programming.

Programmatic Navigation

The ability to navigate the dynamic, "sessionless" state of the web most clearly distinguishes the intelligence of one agent from another.

The fact that each agent request for a web page is a discrete and discontinuous event with no relationship to the previous request has resulted in some clever workarounds. One of these is cookies, a few lines of code saved to the browser's hard drive that uniquely identifies the user. Cookies provide an elegant, economical way for a server to recognize a browser over time and are sometimes required as part of the server's security protocol.

Another means of maintaining state is a unique identifier assigned by a web server to each browser that visits. The identifier is appended to the URL of any page requested by that browser (or hidden in an invisible form passed from page to page), providing continuity between page requests, but the unique identifier persists only as long as the browser is active on that site. Returning precisely to the same data over time may require that the agent duplicate the session variable.

And, of course, a user may be required to identify themselves before being admitted to a site. The challenge/response protocol of username/password is commonly used to guard proprietary or paid content on the web.

Physical pinpointing is quick and dirty. Logical pinpointing, while more demanding, is also more robust.

The navigational ability of an agent distinguishes its intelligence.

The programmatic navigation of the web may require that the agent duplicates cookies, server variables, and username/password. And sometimes it requires the agent to intentionally remove all traces of its visit.

The Invisible Web

The invisible web or deep web is completely opaque to search engines.

From a business perspective, much of the really valuable information on the web – product pricing, inventory, descriptions, schedules, tabular data, management profiles, patent registrations, government mandated disclosures – is stored in databases and accessible only after filling out and submitting a form.

An example: a number of sites on the web offer unique information about particular genes, each identified by its gene sequence. Sequences are similar to reference numbers and are required as form data submitted to a database in order to retrieve each site’s unique information about that gene. Many of these sites also have calculators to compute things like thermal dynamic properties. Calculators also use the gene sequence as input.

Because this data is inaccessible to search engines, it’s sometimes called the “invisible web.” An intelligent agent, however, can submit each gene sequence of interest, follow dynamic links generated in response, re-submit the form based on the data returned, extract the relevant data, and export it to a database or spreadsheet for further analysis or redistribution.

An agent may need to replicate all of these variables – cookies, session variables, username/password – as well as iteratively populate and submit forms in order to extract the desired data.

Popular File Formats

An agent’s fluency with different file formats increases its utility.

The web is a vast network housing numerous popular file formats other than HTML/XML. Adobe Acrobat (PDF) files have become as ubiquitous as Microsoft Office files (Word, Excel, and PowerPoint). Increasingly publishers are incorporating text in graphic images, requiring Optical Character Recognition (OCR) on the part of the agents.

Extending an agent’s reach inside the corporate firewall requires the ability to read email archives (POP3 and IMAP) and RSS feeds.

The ability to query ODBC-compliant databases enables an agent to integrate external and internal data without the use of web services or COM objects.

Treading Lightly

Poorly programmed agents can consume web server resources.

External sources such as public websites should be used discreetly and with care to tread lightly on server resources and bandwidth. Used maliciously or even carelessly, the power of intelligent agents can overwhelm a web server. Any agent that queries external repositories should include the ability to throttle it’s own page requests, controlling the load imposed on remote servers.

Gentle Art of Self-Defense

An agent programmed to gather competitive intelligence may need to defend itself from counter-intelligence.

There are perfectly legitimate situations where you may not want to leave any footprints – such as an IP address – in a competitor’s web log. This has long been true in the competitive intelligence arena where counter-intelligence techniques have evolved to prevent the robotic harvesting of publicly available data. Of course, the data can still be retrieved manually – it is, after all, publicly available – but the idea is to make it more difficult to incorporate the intelligence into competitor’s automated processes and consequently less useful. The ability to cloak an agent’s identity or conceal its point of origin is a useful option.

We’ve addressed the qualities that make intelligent agents so useful but how do you employ an agent? How do you begin taking advantage of the technology?

Building Intelligent Agents

Basically, there are two paths to building intelligent agents that automate your data collection: custom programming or Integrated Development Environment (IDE). Either may be appropriate, depending upon your application.

Custom Programming

There are custom software shops that program ad hoc agents. Custom programming, however, can quickly become an expensive solution. Given the challenges that agents face in the field, hiring programming talent with the experience to build adept agents isn't cheap.

Scalability can also be an issue. When an enterprise discovers the utility of employing intelligent agents for external data collection, it's appetite for information can become voracious. The sheer horsepower of the solution must keep pace. TCP/IP processing, error handling, data file formatting, all need to occur transparently. Massively parallel deployment engines, multi-threading and load balancing across multiple servers may be required.

Some questions to address when considering whether custom programming should be done in-house or out-sourced:

1. If the object is to gather sensitive competitive intelligence, how secure is your data? How protected is your identity?
2. What is your time-to-market? How much time will it take to maintain?
3. If the data proves useful, can you quickly scale production? At what cost?
4. Are you re-inventing the wheel? Is the desired functionality available off-the-shelf?

IDE

An Integrated Development Environment includes everything you need to build, deploy, and debug an intelligent agent. IDEs can be either programmatic or point-and-click.

Point-and-Click

Point-and-click IDEs typically consist of wizards that guide the user through a linear progression of choices with successive options dependent upon previous choices. The idea is that non-programming staff can create effective agents at less expense than IT staff.

The user's guide for one point-and-click IDE package recommends a "basic understanding" of programming constructs, debugging, HTML/XML, and scripting languages such as JavaScript, VBScript, and regular expressions – hardly the skill set of most non-technical staff.

"Basic" may be disingenuous given the complexity of navigating and identifying data on the web, de-constructing conditional logic in client-side script, or understanding changes in a web page that have invalidated an agent's programming. In reality, point-and-click software often disguises the complexity of the problem rather than simplifying the solution.

While its true that staff with no programming experience may be able to build rudimentary agents with simple missions, some web programming skills are required for anything more complex.

Point-and-click IDEs actually tend to penalize expertise, segmenting the task into very small chunks and preventing a holistic view of the solution. This fragmented approach requires even an experienced developer to deal with each small segment of the solution sequentially through the IDE interface rather than confronting the code directly.

Programming of intelligent agents must scale effortlessly to meet enterprise demand.

Point-and-Click development can quickly create rudimentary agents for simple missions.

“Use of SQL reduces the learning curve for most analysts...”

Programmatic

An alternative to point-and-click development leverages SQL (Structured Query Language) syntax to query unstructured data sources. SQL is a powerful declarative programming language with a concise syntax. It has the added advantage of a broad user base, reducing staffing cost and availability issues.

Employing a SQL-like syntax allows the developer to use nested queries and views from various data sources, restrict data sets using where and having clauses, test against case and if statements, group and order data sets, iterate using arrays, manipulate data with string and numeric functions, and match patterns with regular expressions.

In a report entitled Commercial Information Technology Possibilities written for the Center for Technology and National Security Policy, the author states, “Use of SQL reduces the learning curve for most analysts by not requiring mastery of computer languages that support artificial intelligence-based approaches to symbol manipulation. The strategy is to pull from a variety of source documents that are hosted on the Internet or maintained by subscription services or internal network servers. This data, once located, pinpointed, and extracted, is then repurposed by converting and storing it in a format of use to the client, such as a spreadsheet or graphic.”

Point-and-click development makes sense in situations that don’t require the power and flexibility of a declarative language similar to SQL or when programming staff are simply unavailable. In those situations, however, a hosted service should also be considered.

Installed Software or Hosted Service?

Installing software or using a hosted service may ultimately depend upon the sensitivity of the data.

In some applications, especially competitive intelligence applications, security of the data is a primary consideration. Even the nature of the query and the targeted source may be a security issue. In such cases the building and maintaining of intelligent agents – and the data extracted – is typically housed entirely behind a corporate firewall. This requires software to be installed on corporate servers, programming staff, and the IT infrastructure necessary to support the application.

In other situations security is less of an issue than the availability of programming staff and IT support. A hosted service that requires only a browser and Internet connection to access the data may be an attractive alternative.

Many of the manufacturers of intelligent agent software offer only one option or the other. The ability to migrate between options provides greater flexibility as business requirements, IT staff, or the competitive environment changes. It may also make sense to incubate agents on a vendor’s hosted service before bringing it in-house.

And don’t forget to ask if the company offers a pre-packaged solution that will meet your requirements off-the-shelf or with some custom tweaking.

Agents for Hire

Set high expectations for your data collection agent.

QL2 is in the business of building intelligent agents for data collection. WebQL, our flagship product, represents a feature set you would expect of a robust IDE for intelligent agents:

- Ability to navigate, extract, and reformat information from both external and internal sources
- Support for commonly used file formats including HTML/XML, Adobe Acrobat (PDF), POP3 and IMAP email archives, RSS, Microsoft Office files (Word, Excel, PowerPoint), RSS feeds, dBase and ODBC-compliant databases
- Optical Character Recognition (OCR) capability
- Concise notation based on ANSI SQL standard
- Massively parallel deployment engine

- Load distribution across multiple servers
- Support for cookies, agents, frames, tables, and authentication
- Form discovery and completion
- Identity protection (anonymization)
- Robust APIs for Java, SOAP, VBA ActiveX/COM, .NET, & C++
- Error handling
- Powerful network monitoring tools
- Page change detection and reporting
- Page request throttling
- Query execution scheduling
- Available as installed software or hosted service
- Integrated Development Environment (IDE)

More Info?

The simple fact is that automating the extraction of data from unstructured sources such as web pages, PDF, RSS feeds, and Microsoft Office files can not only provide your company with a competitive advantage, it can save you a ton of money.

Interested in more information about QL2's agent technology? Visit www.QL2.com and request a **free demonstration**.

sales@QL2.com