IDN
Internet Draft
<draft-ietf-idn-dnsii-trace-01.txt>
Intended Category: Informational

Edmon Chung & David Leung
Neteka Inc.

July 2002

Transitional Reflexive ACE û IDN Transition (IDNX)

STATUS OF THIS MEMO

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of RFC2026 except that the right to produce derivative works is not granted.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts. Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The reader is cautioned not to depend on the values that appear in examples to be current or complete, since their purpose is primarily educational. Distribution of this memo is unlimited.

The list of current Internet-Drafts can be accessed at
http://www.ietf.org/ietf/1id-abstracts.txt
The list of Internet-Draft Shadow Directories can be accessed at
http://www.ietf.org/shadow.html.

Abstract

This document, while sharing a similar concept, is an overhaul of TRACE-00 and describes a strategy for domain name server operators to prepare and transition their services for multilingual domain names (IDN û Internationalized Domain Names).

The TRACE-01 or IDNX (IDN Transition) approach accepts that users with non-IDN aware applications will be attempting to access multilingual domain names. Hence it is the registry or domain operator's responsibility to provide an IDN solution that resolves these issues to make it a seamless transition experience for technically unsophisticated users.

In essence, the IDNX approach embraces a complementary server-side implementation to smooth the transition. IDNX ready domain servers should be backward compatible, and successfully resolve IDN requests sent via non-IDN aware applications, whether they are formatted in local encoding, UTF-8 or an identifiable variant; as well as forward compatible, and be able to resolve ACE requests.

The IDNX approach also utilizes a dynamic CNAME mechanism (similar to TRACE-00) to provision for the sub-delegation of multilingual domain names to hosts running non-IDN aware DNS servers.

Table of Contents

1. Introduction

During the lengthy discussion at the IETF IDN workgroup, the question about time-to-deployment is often being raised and considered a critical concern. However little word was said on any actual transition path towards IDN. This document provides a comprehensive guide to the issues surrounding the deployment as well as information on resolving them during the transition period.

This document is intended to be an informational paper offering suggestions to implementers of IDN and registries wishing to provide IDN registration to make it a more seamless transition for end users. IDNX is a transitory approach and is complementary and compatible to the eventual standard protocol for IDN.

In fact, some or all of the aspects and suggestions discussed in this paper has been implemented at various domain registries already. The extent of coverage differs from the different implementations. One key observation however is that all are compatible and cause no damage to the Internet at large.

1.1 Terminology

The key words "MUST", "SHALL", "REQUIRED", "SHOULD", "RECOMMENDED", and "MAY" in this document are to be interpreted as described in RFC 2119 [RFC2119].

1.2 Assumptions

This document assumes that the reader has basic knowledge of the DNS and IDN. For more information on the IDN discussions and background knowledge checkout: http://neteka.com/en/pages/products/position.html

2. Common Misconceptions

There is a common misconception among domain registries and registrars that once a proposed IETF standard (in the form of an RFC: Request for Comments) for multilingual and Internationalized domain names (IDNÆs) is published; Registries will immediately be able to accept registrations and deliver working multilingual domain names to end-users. In fact, the publishing of RFCs for IDN is but the beginning of a long transition towards a common and universal multilingual namespace.

Another common misconception is that a server-end approach would cause a lengthier transition. That however is only based on a server protocol overhaul for IDN and not the use of a server-end transitory solution. This document specifically describes a solution for the transition and utilizes a complementary server-end approach to smooth the move with the end user in mind.

3. The reality of a Purebred IDNA Approach

Simply put, ACE is a process whereby multilingual domain names are converted into an alphanumeric string containing only A-Z, 0-9 and hyphens "-" (e.g. bq--ad9ebbvfnqerv.example). IDNA mandates that this transformation take place before an application, for example the browser, sends a DNS request over the Internet (over-the-wire) to a registry name server.

Success of this "Client" only approach therefore has two inherent pre-conditions;

1. Firstly, in order for IDNA to function universally, all software programs that interact with a domain name MUST be upgraded with the ACE standard (browsers, email applications, html editors, audio/video streaming applications, word processors, operating

system tools, ftp agents, etc.). Or the user will have to type in the ACE string to reach multilingual domains; and

2. The success of IDNÆs will depend on the successful propagation of those client software, which will be highly dependant on a) the successful dustribution of client plug-in in the short-term and b) the incorporation of the standard in next generations of application software (e.g. MSIE and MS Outlook)

3.1 Lengthy Transition & User Confusion

A key concern with respect to a purebred client approach to IDN is that there are simply no efficient or effective means to distribute the client plug-ins.

Even if we assume that a dominant software provider, such as Microsoft, with their penetration into the browser and email client market, was to introduce a new version of its Internet Explorer and Outlook line of products with the IDNA client built in, it will still take a considerable amount of time before these new versions reach the majority of desktops around the Internet.

Using Microsoft IE 6.0 as a yardstick for new browser adoption rates, which yielded an eye-popping 34% penetration within 7 months of its release, the industry is still effectively looking at a minimum of 24-36 months or more before a critical mass of Internet users will have adopted an IDN standard-ready-browser. That is not taking into consideration the lead-time for Microsoft to release the next version of their software, let alone whether IDN will make the cut for its next version, and that the IDNA components are stable. This will realistically add another 12 months to the process.

Adoption rate from different regions however may be different and correlated to the urgency of having a complete IDN system in place. For example, regions where non-Latin based character sets may have more adoption faster. However early set backs by domain providers who have not addressed the transitional issues for the user may cause loss of confidence by users of multilingual domain names.

Finally, the client side approach to internationalized domains is especially susceptible to client plug-in software conflicts as providers scramble to push out these plug-in software applications. Client plug-in providers may purposefully or inadvertently interfere with how the browsers handle multilingual characters causing greater confusion.

To summarize, the purebred IDNA approach requires no changes to the DNS infrastructure but does require fast and efficient distribution of client software that supports the standard, which is difficult to achieve. Registries that cannot provide reasonable access to IDNs from the start will likely experience support headaches and many dissatisfied users who are confused and frustrated because their multilingual domain names do not work. Further, without general accessibility, the value of multilingual domains will be compromised, ultimately causing the move towards IDN to be even slower.

3.2 Registry Responsibilities

Because the average Internet user is not expected to be technically sophisticated and know that they have to upgrade their existing applications (such as the browser) before using multilingual domain names, it is the opinion of the authors that it becomes the registryÆs responsibility to provide a multilingual domain deployment that is transparent to its users.

In other words, the registry should be prepared to accept and resolve multilingual domain requests from users with existing software when they deploy and offer multilingual domain registrations to registrants.

Of course, individual policies for each registry may differ. This document intends to offer some suggested solutions to domain registry operators in dealing with real life issues about the deployment of multilingual domain names.

4. Complementary Server-end Brute Force Resolution

We understand that it will be a lengthy transition to IDNA given that it requires upgrade of millions of clients and applications that are already out there. IDNX complements it with a solution that can be deployed at one single node (the registry) and make it possible for the millions of existing users and hosts to immediately be able to access and use multilingual domains.

4.1 The IDNX Premise

The premises of IDNX are:

1. To support and complement the adoption of the open IDN standard;
2. To present an option for a working solution that provides a seamless transition to the IDN standard;
3. Be fully compliant with the IDN standard by preparing the registry for multilingual domain requests sent via existing applications as well as upgraded IDN aware applications;
4. Accepting and addressing the fact that multilingual domain requests will be successfully sent to the registry from existing software and will resolve those requests instead of avoid them;
5. To provide a transitory solution that will make multilingual domain names functional immediately;
6. To allow the host for a multilingual domain name to use an existing (BIND, etc.) DNS software by using only an ACE record (only registry Servers need be updated or supplemented);

In short, the authors believe that a complementary server side approach to IDN is a crucial component for the successful deployment of IDNs at a registry. This document outlines possible approaches to the resolving of an IDN and how IDNX can both accelerate adoption and reduce the risk of deployment of the IDN standard for registries.

4.2 Resolving Domain Names

In general, there are three types of IDN requests that a registry server may receive: 1. Domain in the form of UTF8 encoding; 2. Domain in the form of its local encoding (GB, Big-5, JIS, KSC, ISO8859-1..13 etc.); 3. Domain is in an ACE (ASCII Compatible Encoding) format. Each of these types includes both the possibility that the received request is in its pure form or that it is a variant of the type. Because the application (browser or the operating system) was not designed for multilingual domain names, even though it does send out the domain request, it often tampers with the characters before sending, causing the advent of a variant being generated.

4.3 Extent of Resolution

Type 1 requests, where a domain is sent in the form of UTF8 encoded characters, are generally received from more recent versions of browsers and operating platforms. These include Microsoft Internet Explorer 5 and above, Netscape 6 and above and applications based on the Windows 2000 or XP platform. These applications were designed to deal with multilingual characters using Unicode, however, because it was not specifically programmed for IDNs, it sometimes malfunction when presented with such. The result is that a variant of the original UTF8 character is formed and sent to the DNS.

Type 2 requests, where a domain is sent in the form of local encoded characters (GB, Big-5, JIS, KSC, ISO8859-1..13, etc.), are generally received from earlier versions of browsers and operating platforms. These include Microsoft Internet Explorer 4, Netscape 4 most other applications based on the Windows 98 or earlier platform. These applications were not designed to deal with multilingual characters using Unicode at all and therefore will be sending out the IDN in its local encoding. Mostly, the domain is also not tampered with before sending; therefore a pure local encoding may be received. However under some platforms and circumstances, it will also try to reinterpret a multilingual domain name. The result is that a variant of the original local encoding is formed and sent to the DNS.

Type 3 requests, where a domain is sent in the form of ACE encoded characters (RACE, DUDE, UTF5, AMC-Z, etc.), means that the application has been upgraded with an IDN client. Because of the evolving standard, depending on the version or provider of the client, the type of ACE string may be different. Additionally, these client plug-ins are susceptible to buggy code, as they are being created as beta products by providers experimenting with the evolving IDN standard. Misbehaviors caused by browsers and operating systems, which are not taken into consideration by the client software also gives way to the creation of variants in ACE formats.

Even though variants are created as a derivation form the pure form, these variations are consistent and predictable, and therefore can be resolved definitively and uniquely. There is no guessing involved. An IDNX server could be designed to handle both requests sent in its pure form as well as its variants from the three types of multilingual domain name requests. The extent of the coverage is only dependent on the extent of research an implementer has put on the subject and their understanding of the IDN issues.

4.4 Ambiguity Resolution

Another concern is for character encoding conflicts between the various 8-bit encoding schemes (including UTF8 and local encoding schemes). IDNX suggests the handling of these issues at the point of registration. On the registration side, the domain name is definitively registered using Unicode, regardless of whether Unicode or local encoding was used as input. The desired domain name is confirmed with the user at the interface, ensuring that the name registered is accurate.

During the availability search, encoding conflict is checked to ensure no existing domain name caused an encoding conflict with the requested domain. In essence, it SHOULD maintain uniqueness of a codepoint string (in its raw hex values) regardless of its original encoding scheme. The system SHOULD reject domain names on a first come first serve basis for domains that will cause conflict with any local encoding or UTF8 string (or variants if applicable).

By taking care of both the domain registration and the resolution side at a domain registry, IDNX can ensure a complete solution and extensive coverage for multilingual domain resolution right from the start. Not only will users with IDN aware plug-ins be able to access the multilingual domain names offered by registries or domain operators with IDNX, users with existing software, existing browsers will also be able to successfully access these names, making it a truly seamless and transparent transition.

4.5 Complementing IDNA (OPTIONAL)

Note that this is an optional feature that can be deployed independently with other parts of the IDNX suggestions, this particular functionality could be omitted without losing the other benefits of deploying the IDNX solution.

To complement the propagation of IDNA clients, IDNX servers could also improve the download experience by dynamically prompting the user for download when users first try to access a multilingual domain name. Because the IDNX solution includes a server-end deployment, the server could be configured to act as a channel for the IDN client at the point of access to multilingual domain names.

In essence, with an IDNX capable server, the registry would be able to successfully intercept multilingual domain requests and offer the user the option to download an IDN client plug-in. If the user chooses to download the plug-in, then the domain would be resolved using the ACE format. If the user declines the option to download, IDNX, preserving the seamless resolution experience for the end-user, would still resolve the domain.

The implementer should pay special attention in deploying this mode of operation for an IDNX server. For example, the implementation should avoid prompting the user too many times for the download. There are three precautions that may be helpful:

1. Before prompting for download, ensure that the user do not already have the client plug-in;
2. Cache the user and refrain from prompting again for a given period of time; and
3. Since the download will likely make use of the http server, setting up cookies may provide a solution for achieving 2.

More specifically, the implementer should take into consideration the end-user and try not to annoy and inundate them with download options.

Finally, note again that this is an optional feature that can be independently deployed.

5. Delegating Multilingual Domain Names

Besides being backward compatible with existing client side applications, the IDNX solution also intends to make it backward compatible for name servers hosting delegated IDNs, as well as for resolvers in the path of resolution.

The biggest challenge, aside from resolving the IDNs, for a registry in rolling out multilingual domain names is how domain registrants could host it. IDNX suggests a Dynamic CNAME mechanism, which when deployed at the registry level, the delegated domain hosts could simply use their existing name server (BIND, etc.) to load in the long term ACE as the domain. It is also possible to further delegate sub-domains to other hosts, just like with the current English only domain names.

5.1 Dynamic CNAME

The ability to sub-delegate multilingual domain names to ACE only hosts is achieved by using a dynamic CNAME mechanism (similar in concept but different in practice to TRACE-00) to glue the multilingual domain name to the long term ACE equivalent at the registry DNS server.

The existing STD13 DNS servers do NOT support this feature. A registry operator or zone administrator MUST upgrade their name servers to an IDNX ready system if they wish to use Dynamic CNAME for multilingual domain transition. The main reason for not using DNAME or another form of resource record is that existing resolvers readily support CNAME and not the others, thereby ensuring backward compatibility. Furthermore, the use of a wildcard CNAME record also allows Dynamic CNAME to be DNSSEC compatible (This will be further discussed in Section 7).

The following is a brief explanation on how the dynamic CNAME should work:

```
- Application sends www.<ML>.example to resolver
- .example Registry responds with:
     www.<ML>.example   CNAME   www.xx--ACE.example
- Resolver then uses www.xx--ACE.example to query the host
- Therefore the host will only have to setup the xx--ACE.example zone
```

With Dynamic CNAME, the registrant for a multilingual domain name will be able to host their domain with their existing DNS server software and be able to create English sub-domains from it (e.g. ldh.<ML>.example).

If multilingual sub-domains are desired (e.g. <ML>.<ML>.example), then the registrant could opt to also make their system IDNX capable.

This transitional strategy could also be deployed at the root level so that multilingual TLDs could be issued and be backward compatibly functional. Therefore, full multilingual addresses could be used (i.e. <ML>.<ML>.<ML>).

5.2 Setting up a Multilingual Domain Name at the Registry

The concept of Dynamic CNAME is relatively simple, imagine the multilingual domain name being setup at the registry as:

```
*.<ML>.example   IN CNAME    *.xx--ACE.example
xx--ACE.example  IN NS       ns1.xx--ACE.example
```

Note again that you must upgrade to an IDNX ready DNS server in order for this setup to work. Existing STD13 name servers (BIND, etc.) do not readily support setting up of wildcard CNAME records as such.

An IDNX compatible DNS will therefore successfully provide a requestor the corresponding ACE record of a multilingual domain request via a CNAME response. Because STD13 DNS resolvers support a CNAME RR, the resolution path would be seamless. Note that the response from the registry server (as indicated in Section 5.1) should be "CNAME www.xx--ACE.example" and NOT "CNAME *.xx--ACE.example".

In essence, no matter what the sub-domains may be for the <ML>.example domain, it will be successfully sub-delegated to the host server using its ACE name.

Conceptually, we could also setup the different types of possible multilingual domain requests and variants as follows:

```
   *.<ML-Type1>.example           IN CNAME    *.xx--ACE.example
   *.<ML-Type1Æ>.example   IN CNAME    *.xx--ACE.example
```

```
*.<ML-Type2>.example           IN CNAME    *.xx--ACE.example
```

The actual implementation of an IDNX compatible server however may cause the setup to be different.

For example, the implementation could make the forking of the variants transparent to the zone administrator, so the administrator only needs to provide one definitive record, for example, simply:

```
xx--ACE.example                IN NS       ns1.xx--ACE.example
```

Upon loading the zone file, the IDNX server would identify the xx--ACE.example domain as a multilingual domain and fork the variants out automatically for resolution purposes.

Note that your current DNS server (BIND, etc.) MAY NOT be capable of handling such a setup. IDNX capability MUST be patched or otherwise upgraded and installed into the existing server if this feature is desired.

5.2.1 Character Equivalence Policies (OPTIONAL)

This is an optional possible feature that can be implemented based on the policy of the zone operator / domain registry.

By using the same concept, a registry may also choose to deploy character equivalency policies for multilingual domain names. For example to deal with identical Latin characters such as the capital letter Alpha "A" and the English capital letter "A". Similarly, it could also be applied to the mapping of Simplified and Traditional Chinese characters.

Essentially, the zone operator may setup a Traditional Chinese domain and dynamically CNAME it to the Simplified version in ACE format (or vice versa). The operator again would ultimately determine the extent of coverage. The more equivalency records are added, the more variations would be taken care of, and the more complete the coverage will be, but the fewer names would be available for registration.

For example:

```
*.<ML>.example    IN CNAME    *.xx--ACE.example
*.<MLÆ>.example   IN CNAME    *.xx--ACE.example
```

<MLÆ> denotes a multilingual domain name in an equivalent form. To prevent conflicts, this should be considered like that discussed in Section 4.4 for ambiguity resolution. That is, it should be taken care of at the point of registration, when domain availability is checked.

Take note again that this is just a possible transitory strategy. A more permanent strategy for character equivalency issues may be further discussed in a different context.

5.3 Setting up a Multilingual Domain Zone at the Host

At the host, the multilingual domain name will simply be setup as a regular English only domain name with the ACE record. The host DNS server does NOT need to be upgraded to IDNX.

For example:

```
xx--ACE.example       IN SOA       ns1.xx--ACE.example dnsmaster.xx
                         NS        ns1.xx--ACE.example
                         MX 0      mail.xx--ACE.example
ns1                      A         123.4.5.6
www                      A         123.4.5.7
mail                     A         123.4.5.8
```

As you may see, it is exactly identical to what you would do for setting up an English only domain, and using English alphanumeric characters only.

5.4 Complementing IDNA

Furthermore, because the IDNX architecture for registries allows for the delegation of IDNs in ACE format to non-IDN aware host servers, this again complements the evolving IDNA side standard whereby the hosts do not necessarily have to upgrade their system for multilingual domains.

6. Using Multilingual Domains on other Applications

Because domain names are used in many applications, they must be taken into consideration for the transitional strategy as well. The two more prominent applications are: email and website hosting.

6.1 IDN in Email Addresses

The IDNX approach will support the use of IDNs in email addresses immediately. With a similar concept deployed at the email servers, users will also be able to use multilingual user names along with IDNs for their email address (i.e. <ML>@<ML>.example).

From Section 5.3, note that the MX RR (Mail Exchange Resource Record) could be easily set in any existing DNS server (BIND, etc.) by simply using the ACE equivalent of the multilingual domain name.   It is actually also possible to setup your existing mail server software for email addresses with English user names (i.e. ldh@<ML>.example) by using a domain independent IP setup.

In order to offer multilingualized user names for email addresses, an email server with the IDNX concept incorporated for the management of user names could be used. Similar to the domain situation, because the email applications were not designed to handle multilingual email addresses, some issues may arise. Also, similarly, we should take that into consideration in the design and deployment of an IDNX ready email server, to make the user experience seamless as multilingual names are deployed.

The same three types of scenarios exist for multilingual email addresses: 1. Address arrives in the form of UTF8 encoding; 2. Address arrives in local encoding format; 3. Address has already been converted to ACE before it arrives. For type 1 & 2, variants may also be generated and are possibly caused by the multilingual email address being converted into MIME format. As for type 3, the same causes for domain names apply. That is, caused by experimental software, versioning, as well as lack of consideration for interaction with other applications. Unlike domain names, variants of email addresses are less likely caused by malfunctioning of the software agent during transport but rather as an effect by design.

Therefore, to successfully provide comprehensive coverage for immediately usable multilingual email addresses before massive distribution of client side plug-ins is achieved, we should incorporate in the design, consideration for all these variants, sharing a similar strategy as IDNX.

Furthermore, IDNs within the email headers could be inscribed in ACE format. This inscription could take place at the SMTP server to allow the user to send the email using their existing MUA. Again, this strategy would further promote and complement the IDNA approach.

6.2 Hosting Websites based on IDN

Besides hosting the domain name (as discussed in Section 5.3), under an IDNX registry, registrants can also use their existing web servers to host websites for multilingual domain names.

To host the multilingual domain name site using an existing web server, the user should configure their web server for IP hosting. In order to offer virtual name hosting however, because it involves the configuration of multilingual domain names, users should elect to use a multilingual enhanced web server based on an IDNX type concept.

That is, having the web server capable of accepting the three types of possible requests: 1. Http requests in the form of UTF8 encoding; 2. Http requests in the form of local encoding; 3. Http requests in the form of ACE. For 1 & 2, there are possibility that the domain name request comes in an http escaped form (i.e. %XX%XX) as well as a variant of the pure form. The same variances for 3 applies.

6.3 Other Known Issues about using IDN

No matter how multilingual names are deployed, a set of problematic glitches would arise as the transition takes place and as users learn to understand more about these issues. The three main reasons being that: 1. The average user is unaware of these problems; 2. System administrators may arbitrarily block multilingual requests by setting up their system that mistake IDNs to be malicious requests; and 3. Existing software was not originally designed for IDNs.

For example, some browsers simply block all entry of multilingual domain names, others try to implement some form of transformation causing loss of character information, which is may be irrecoverable. The DNS resolvers residing at the ISP level are very important "messengers" in the DNS. Since the original DNS protocol itself is arguably 8-bit capable, this middleman usually just pass requests along the DNS path, however proxy and caching issues could complicate matters. Proxy servers and cache servers will contribute to the blocking of multilingual names, by misinterpreting them as malicious requests or other undesired traffic. Consequently creating a barrier for multilingual names to be successfully transported.

Multilingual email addresses also have its share of issues. There are three main areas where issues may arise: during the sending of emails through the SMTP servers; the retrieval of emails from the POP servers; and client side blocks. For example, some existing MTAs equipped with ASCII-check statements may simply deny receipt of messages from a multilingual address, while others equipped with virus scanning features, may reject non-ASCII characters in an e-mail address misinterpreting them as potential malicious attacks. The interfaces of some email user agents as well as mail portals are equipped with ASCII-check statements, blocking all attempts to send messages to multilingual email addresses. The major email clients however will be able to manage multilingual email addresses when setup correctly.

While the authors admits that there will be some stubborn known issues surrounding the deployment and usage of multilingual domain names, the IDNX strategy suggests a comprehensive solution for a registry in the deployment of multilingual domain names to substantially reduce complications brought by technical issues as well as for customer support issues.

7. Security Considerations

Because DNSSEC provides coverage for wildcard records, the Dynamic CNAME feature should not cause any additional security concerns.

Besides the above, this document does not talk about DNS security issues, and it is believed that the proposal does not introduce additional security problems not already existent and/or anticipated by adding multilingual characters to the DNS.

References

[IDNA] Internationalizing Domain Names in Applications, draft-ietf-idn-idna, Feb 2002, Patrik Faltstrom, Paul Hoffman, Adam M. Costella

[PUNYCODE] Punycode: An encoding of Unicode for use with IDNA, draft-ietf-idn-punycode, Feb 2002, Adam M. Costella

[STRINGPREP]Preparation of Internationalized Strings, draft-hoffman-stringprep, Feb 2002, Paul Hoffman, Marc Blanchet

[NAMEPREP] Nameprep: A Stringprep Profile for Internationalized Domain Names, draft-ietf-idn-nameprep, Feb 2002, Paul Hoffman, Marc Blanchet

Acknowledgements

The authors would like to take this opportunity to thank all those who have provided precious comments and feedback during the preparation of this paper.

Special thanks to the following persons whose comments have been invaluable to this document:

Mark Davis
Eric Hall
Tedd Stirling
Maynard Kang
Deng Xiang
Prof. L. M. Tseng
Liana Ye
Martin Duerst
Dan Oscarsson

Authors:

Edmon Chung
Neteka Inc.
Suite 100, 243 College St. Toronto,
Ontario, Canada M5T 1R5
edmon@neteka.com

David Leung
Neteka Inc.
Suite 100, 243 College St. Toronto,
Ontario, Canada M5T 1R5
david@neteka.com