NA3 Investor Prospectus

The NA3 platform encompasses both server-side components (the NCN, or "Native/Cloud-Native" framework) and client-side components (the A3R, or "Application-as-a-Resource" framework). This paper will discuss how Linguistic Technology Systems, Inc. (LTS) envisions marketing and productizing elements of both frameworks, so as to define an implementation strategy and business model for NA3.

The current document will focus on NA3's default implementation via the QT platform. QT is the most widely used native cross-platform application development framework, with approximately one million active developers, over 5,000 client companies, and tens of mullions of downloads of recent QT versions (metrics according to the QT company). Companies employing QT form a natural customer base for QT. The QT-based market, however, is even more substantial than these figures indicate, for the following reasons:

- QT is a commonly used platform for embedded systems, touchscreen devices, Cyber-Physical devices, and other User Interface environments apart from conventional desktop software, with particular prominance in such sectors as aeronautics, automotive, or military cockpits or consoles, wearable devices, biomedical equipment, and hybrid mobile/desktop applications. Most of these technologies depend on networking for data management and persistence, because embedded systems (in comparison to native software on desktop computers) have limited CPU and file-system capabilities.

- Despite the significant market presence of QT in the area of User Interface front-ends for embedded/touchscreen systems, there is not a comparably well-developed QT server or Cloud-Native ecosystem. The QT company launched "QTCloud Services" in 2013, but discontinued this project several years later. The QT platform provides convenient client libraries for Cyber-Physical and web networking protocols such as HTTP, MQTT, and CoAP, so QT front-ends can be implemented to network with generic web or IoT servers. Nevertheless, many QT developers have expressed a desire for an integrated environment where QT tools and protocols are usable on both server and client sides.

  In the context of NA3, the most direct use-case for applications managing embedded systems and/or Cyber-Physical data would be integrated solutions which include desktop software for realtime or post-hoc access to and analysis of Cyber-Physical/IoT information. NA3 is implemented to prioritize destop clients, but is also quite applicable in the case of "triple-endpoint" hybrids which integrate embedded devices that generate raw data, servers which receive and store this data, and desktop cliets which manage the data.

- QT is the basis for one of the two main User Interface technologies employed on Linux systems (the other being GNome), so a substantial percentage of Linux users run software on QT (often without realizing as much). Consequently, any software implemented to target Linux/QT Operating System environments (such as KDE) is a viable candidate to benefit from NCN back-ends or A3R components.

- QT is used for many scientific computing applications, in many disciplines; a representative sample includes TeXstudio (for LaTeX processing), Mendeley Desktop (for Reference/Citation

Management), CERN ROOT (CERN's physics/subatomic analytics platform), IQmol (for chemistry/molecular physics), medInria (for radiology), QGIS (for Geoinformatics), MeshLab (for 3D modeling), OpenSCAD (for 3D geometry), Octave (a MATLAB emulator), ParaView (for data visualization), and the QT Creator IDE (Integrated Development Environment). Most of these applications would not be included in the QT company's official user metrics because they are maintained by academic or research institutions with an open-source QT license; nevertheless, institutions allocate resources for keeping their technical applications up-to-date with current computing trends. This kind of scientific software provides natural targets for integrating desktop applications with NCN back-ends. Similarly, NA3 would be a useful toolkit for implementing new technical software driven by scientific innovations.

Having emphasized use-cases in the QT market, it is worth adding that the unique technical innovations expressed via NA3 have applications beyond the QT ecosystem. In most cases, QT data types and protocols have corresponding equivalents in other application frameworks, both cross-platform and Operating-System-specific, such as wxWidgets (cross-platform), Xcode (Apple) or MFC (Microsoft Foundation Classes). A reasonable estimate is that porting NA3 to non-QT platforms would comprise a 6-month project for a two- or three-person development team.

In order to stay focused on currently-implemented prototypes and the near-term business model, however, the remainder of this presentation will restrict attention to the QT-specific version of NA3.

NA3 Revenue and Marketing

History suggests that most commercial software products generate revenue, in their early stages, primarily from customer-specific customizations, but then eventually derive their most valuable profit-stream from commercial licenses. Special-license customizations help mold the product into a widely-usable standard version, given the natural feedback loop which emerges as the product's development team implements project-specific deployments, obsereving "in the field" which features are most useful and how these features are best made available to developers.

Taking the official "QT partners" cohort as a representative cross-section of the QT market, we can find companies whose revenue is driven by custom software development (ICS, Woboq, KDAB, Base2, Bitfactor, Sequality), by commercial licensing (Wind River, VNC Automotive, FrogLogic, NXP), by realtime and/or platform services (Mender, Timesys, Mapbox), as well as hardware/mircroprocessor providers (Toradex, ARM, Texas Instruments). It is probably true that companies whose products depend on QT, in whole or in significant part, generate revenue from customization and consulting more than in generic technology sectors. This may reflect the position of front-end technology in relation to software in general: many software project begin with new kinds of data or new user-interaction models, and only later address the need for implementing high-quality GUIs. Given that desktop-style front-end development is a rather specialized subdiscipline, many companies end up hiring QT-focused companies as service contractors, which in turn supports a robust ecosystem of QT partner companies. Having said that, financial records released by the QT company itself suggest that commercial ("Developer" and "Distribution") licenses are QT largest revenues source (targets released in 2018 indicate that The Qt Group Plc aims for 60% revenue from licenses, 20% from consulting, and 20% for "support and maintenance" (which is an offshoot of developer licenses).

It is premature to estimate a comparable partonomy of revenue share for **NA3**, but we can identify four distinct profit streams appropriate for **NA3** as an integrated platform:

Customization Custom-implemented applications using project-specific versions of **NCN** and/or **A3R**.

Licensing Commercial licenses required for any deployment of **NCN** outside LTS-controled servers and/or any deployment of **A3R** applications (or of software including **A3R** components for such development requirements as databases, data modeling, scripting, data serializing/deserialization, and text parsing) in a commercial context.

Hosting LTS anticipates running proprietary containers via a Cloud-Native service such as OpenShift, and then leasing access to this service to **NA3** users. Because the expertise involved in building native desktop applications is very different from the techniques required to deploy a Cloud-Native container image, LTS can offer integrated hosting and consulting wherein LTS fully implements and maintains a back-end paired to any desktop/native client software.

Sponsorship As discussed below, LTS anticipates running a data-sharing platform which would be a publicly-visible introduction to LTS's in-house **NCN** service (whereas other sub- or para-containers would be leased to third parties and provide publicly-visible content at their discretion). The "demo" container, while being a vehicle for the general public to learn about **NA3**, would also host research data sets and would therefore be a resource in the public interest, allowing LTS to receive compensation from companies financially supporting the portal because of its merits as a technology benefitting science or scholarship.

For the remainder of this summary, then, the focus will be on cloud services expressly maintained by LTS (in contrast to commercial **NCN** instances whose licensees host the **NCN** code on their own). In **NCN** parlance, sub- or para-containers are units within a larger **NCN** environment that have isolated **HTTP** access protocols and data/file storage. Each such partial-container can be twinned with a specific **A3R** application, providing a cloud end-point for storing application-specific data and/or sharing such data between different executable instances of the application. Potentially, then, and **A3R** project developed by LTS will have a corresponding presence on LTS's cloud resources.

At the same time, the **A3R** (Application-as-a-Resource) model also envisions desktop applications as self-contained, sharrable units, which can be hosted on web servers (including **NCN** instances) as zip and metadata files. Therefore, LTS's **NCN** deployment can serve as an access-point for users acquiring or obtaining information about **A3R** applications (including data sets published as "Research Objects" using **A3R** within their code base).

NA3 and the Research Object Protocol

Open-access data sets conforming to the Research Object Protocol are a good example of use-cases where the **A3R** development strategies may be beneficial. According to the Research Object

Protocol, data sets are paired with code and metadata to help subsequent researchers use and interact with the published data. Via **A3R**, the Research Object can be implemented as a standalone, desktop-style "dataset application" whose data models and **GUI** components are uniquely designed for the associated data set, reflecting its scientific and theoretical provenance (experimental setup, data-acquisition methodology, data-structural rationale, etc.). **A3R** employs unusually rigorous modeling for application components and data types, which makes it particularly appropriate for this Research Object context wherein a data set's technology — its structural organization and custom code base — becomes itself a scientific artifact.