



Analyze | Assure | Accelerate™

TL1 Command Documentation - Proposal and Preliminary Project Plan

**Russell Ward
Spirent Communications
15200 Omega Drive
Rockville, MD 20850**

Sept. 30, 2005

Introduction	1
Historical process	1
Modern authoring/publishing architectures	3
Introducing a modern architecture – Spirent/TL1 style	4
A closer look at a new architecture	7
XML and XSLT – The foundation	7
The XML repository	8
Content contribution (authoring) and editing.....	8
Publishing PDF manuals	9
Publishing “help file” references	10
Publishing a “real-time” reference over the web	12
Architecture summary.....	14
Proof of concept	15
Steps towards implementation	15
Step 1 – Form a collaborative team.....	15
Step 2 – Determine exactly what a TL1 command reference should consist of.....	16
Step 3 – Design the XML DTD/schema for the repository.....	16
Step 4 – Design the publishing processes.....	16
Step 5 – Migrate existing content to XML	17
Step 6 – Determine and establish authoring protocols	17
Step 7 – Deploy and train	18
Roles	19
Challenges.....	19
Migration of legacy content.....	19
Development of a technology-driven infrastructure	20
Gaining acceptance	21
Risks	21
Summary	22

This document represents a proposal for the future authoring and publishing of TL1 commands at Spirent Communications SAB. The purpose is to outline a new methodology for accomplishing TL1 documentation and discuss the path towards achieving it.

Introduction

The TL1 command documentation set provides a major opportunity for overhaul, particularly in the methods used to author and publish it. With more refined processes that engage modern technical communication (techcomm) technologies, Spirent will be able to realize a higher level of output quality, while simultaneously gaining greater efficiency during authoring and publishing. In other words, we can use technology to replace the costs and inevitable errors introduced by human busywork.

The following sections attempt to summarize the current TL1 documentation workflow, and compare it to a proposed architecture that engages higher technologies and less human labor. Afterwards, the remainder of the document will focus largely on the considerations and challenges of implementing such an architecture.

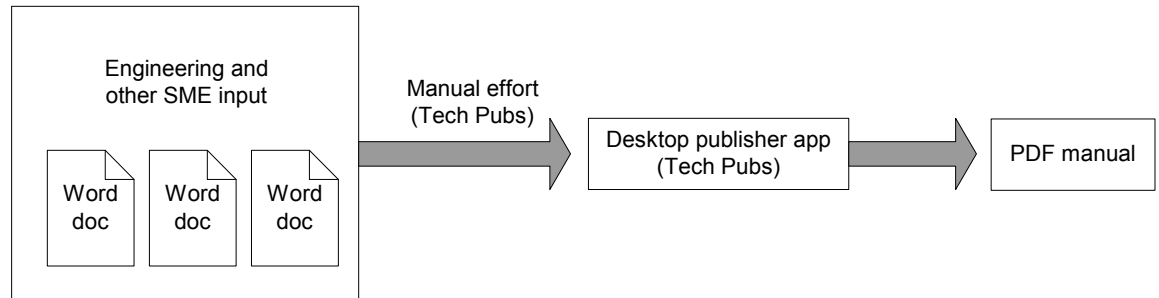
Historical process

TL1 command documentation has historically followed a “traditional” publication process, in that:

- In its desktop publishing (DTP) software, Tech Pubs maintains a command manual “book” for each applicable hardware product, detailing all applicable TL1 commands.
- During the course of software/hardware development, engineering personnel and other subject-matter experts (SMEs) produce rough internal documents in some word processor, typically Word, detailing new and/or changed TL1 specifications.

- The rough documents are forwarded to Tech Pubs, where the updated information is manually merged into the existing content base, with perhaps some editing involved.
- When a GA release occurs, Tech Pubs produces a PDF manual from the DTP software representing the TL1 Command Manual for that release.

A simplified diagram might appear as follows:



Historical TL1 documentation workflow

This approach, while simple and straightforward, has substantial limitations that have resulted in numerous problems:

- The content is essentially authored twice, with the second effort conducted largely through copy/paste effort by Tech Pubs. This redundancy causes significant inefficiency in the process. Furthermore, as with any attempt to manually duplicate a piece of information in any form, errors are inevitably introduced into the duplicate.
- Frequently, a given command will be used by multiple different types of hardware/firmware, creating the need for meticulous sleuthing by the Tech Pubs member when an update occurs. In the DTP repository, each command manual resides in its own “book,” with a complete set of independent information. Again, this redundancy and the manual effort to manage it inevitably introduce errors.
- No established pattern or protocol exists for procuring the source information from the SMEs. Numerous small documents are authored at any given time and float about with no overarching control, raising persistent issues with accuracy, completeness, and versioning.
- The current workflow provides a path into DTP software that is mostly suited for PDF output only, which is the least-favorable format for delivery of this sort of reference information. Even if a more accessible format is a concern for external distribution, the wealth of internal

consumption would be well-served by a more timely and usable delivery format.

- Because Tech Pubs and the DTP software become the aggregation point for all TL1 command information, our full library of TL1 intellectual property becomes locked away in the proprietary format of the DTP. Only Tech Pubs, with its special and expensive software, can access the information repository. This system is contrary to modern concepts of enterprise content sharing, a key component addressed throughout this proposal.
- In the current process, the value-add by all parties is diminished. SMEs and other contributors spend an inordinate amount of time fussing with templates and scattered, uncontrolled source documents that ultimately get thrown away. Tech Pubs personnel are reduced to copy/paste clerical activity, rather than adding real value through publishing and delivery expertise.

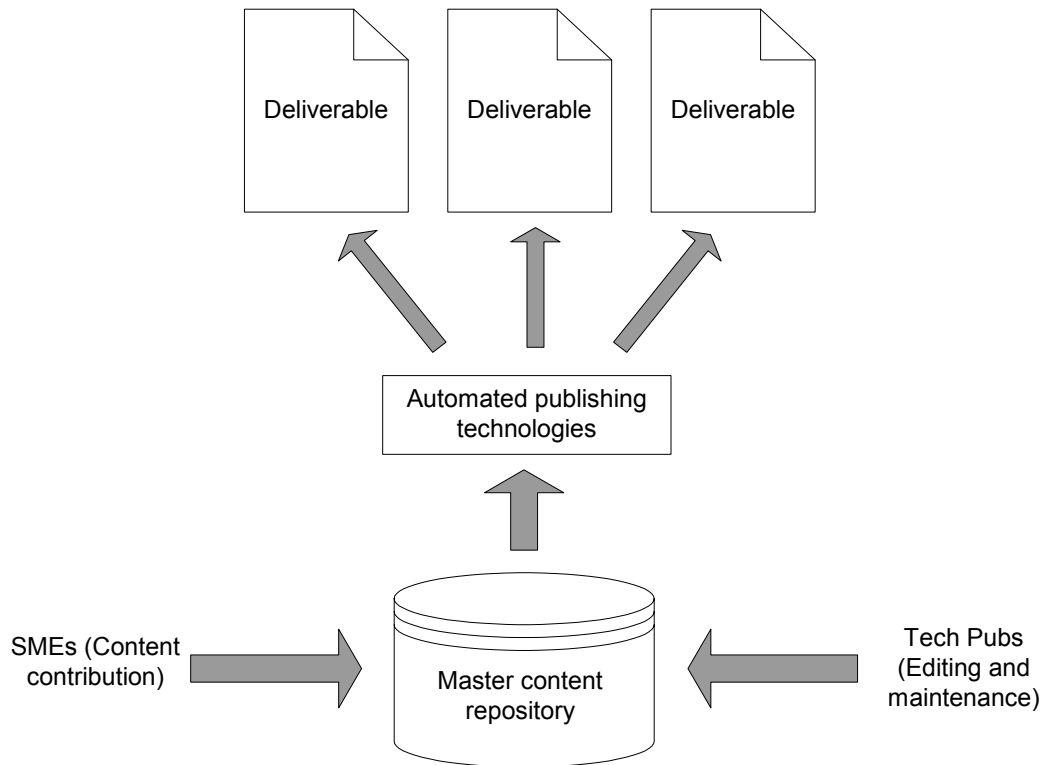
While numerous other limitations and disadvantages exist, these represent the highlights. It should be noted that while the current process is significantly flawed, this paper does not intend to suggest that the creators of the current workflow were negligent. Rather, the main point is that modern techcomm technologies have found ways of improving these processes, and that the time has come to harness the benefits they have to offer.

Modern authoring/publishing architectures

In the previous section, the current workflow was illustrated as a process where disparate sources of information filtered manually through the Tech Pubs writer, who then produced deliverables. This situation, while common in the past, has become obsolete in situations where:

- The primary content contributors work outside of the Tech Pubs department, and
- The primary role of Tech Pubs is publishing, not raw authoring.

A more modern architecture, in a very general sense, would look something like the following:



A more modern architecture

This type of architecture overcomes many of aforementioned limitations because all authoring and publishing focuses on a central content repository. Rather than having engineers writing in some word processor, and Tech Pubs translating into a restricted DTP environment, content is written one time only and published directly from that source.

In the following sections, this model will be expanded to address the specific needs of Spirent, with each component explained in greater detail.

Introducing a modern architecture – Spirent/TL1 style

The key to an architecture based on a centralized repository is using a repository format that:

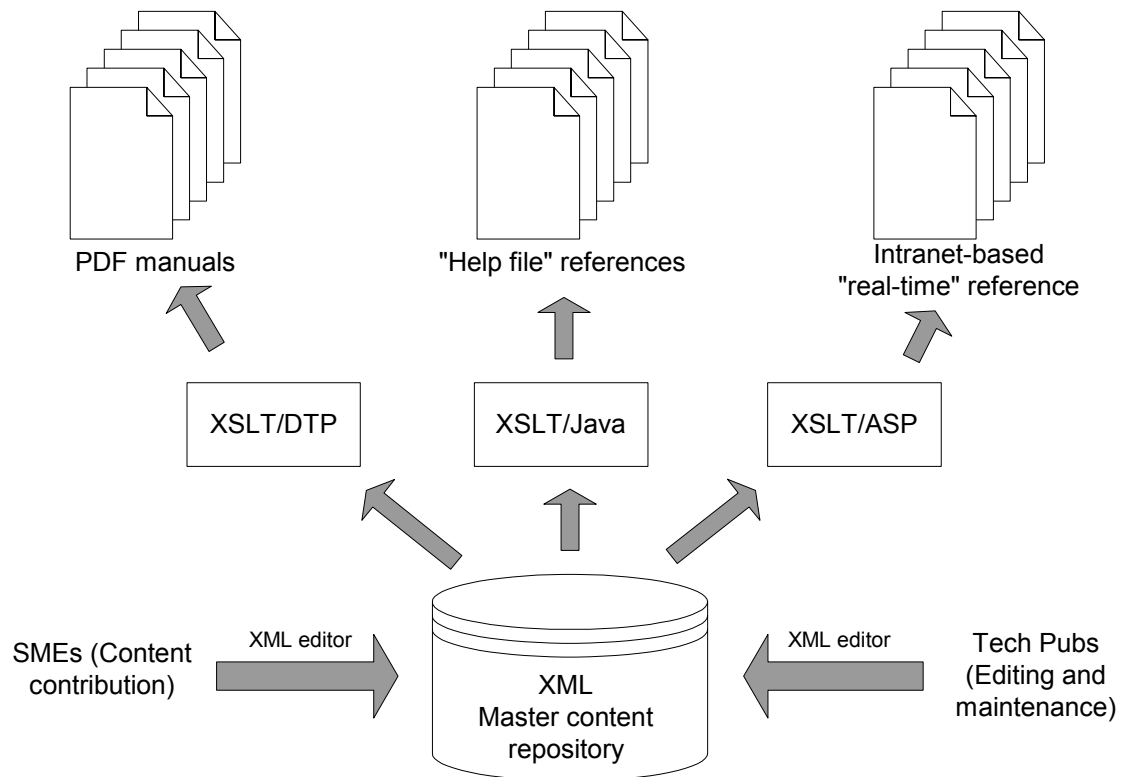
- All potential authors can access

- Supports publishing (PDF, HTML, etc.) from that format, preferably in an automated fashion

In the current industry, the most popular format by far has become eXtensible Markup Language, or XML. XML is an ideal interchange and publishing format for technical content because:

- It is non-proprietary and can be used in any supporting application and operating system.
- Basic tools to generate and manipulate it are inexpensive to free.
- Through transformation technologies, it is highly amenable to publishing in various formats, and the support for divergent formats is steadily increasing.
- The hierarchical model of XML is extremely well-suited for representing literary content, especially technical content.
- For storage, XML is very accessible and flexible. XML repositories can range from simple file systems to complex databases and management systems.

With this established, consider a revised diagram of the modern authoring/publishing architecture, made more specific by assuming an XML format:



Proposed XML-based architecture for TL1 command documentation

Successfully implemented, this architecture would solve all the noted problems with the current workflow, because:

- Content is authored and stored in one place only, with no redundancy or duplication.
- The placement of any given command in the deliverables is controlled by metadata in the repository and parameters of the publishing technologies. In other words, each command appears only once in the master library, and the technology handles it from there.
- Disparate sources are completely eliminated, and clear standards and protocols for authoring can be established through the structural validation qualities of XML.
- XML provides a path to multiple delivery formats, including the possibility for an internal, web-based real-time reference of the current information.
- The aggregation point for TL1 intellectual property is the shared, enterprise repository, which resides in a format accessible to all.

- The process maximizes the value added by all parties. SMEs use their specialized knowledge contribute content, and Tech Pubs uses its expertise to focus on editing, publishing, and delivery. In other words, it represents the optimal utilization of available skill sets.

The following sections will explore the individual components of the diagram in more detail. Afterwards, the document will outline the steps and challenges towards implementation.

A closer look at a new architecture

This portion of the document is intended to explain the components of the proposed new architecture in more detail. Note that technology is a key factor throughout, and the intricate details of these technologies are beyond the scope of this discussion.

XML and XSLT – The foundation

As explained earlier, the foundation of this proposal is the use of XML as a storage format for TL1 documentation content. Throughout techcomm circles, XML is very commonly used for this purpose, for the same reasons it is sought after here. Furthermore, its popularity continues to grow with increased support through associated tools and technologies. Therefore, it represents a stable and low-risk format for Spirent to use for content storage.

The primary technology by which XML can be transformed into a publishable format is through eXtensible Stylesheet Language Transformations, or XSLT. The XSLT standard is maintained by the Worldwide Web Consortium (W3C) and widely used in conjunction with XML-based processes. Among countless other things, it provides a path from XML to PDF and HTML, which are the primary functions of interest in this document. These functions of XSLT are well-supported and tested throughout the techcomm industry. Many tools that perform XSLT functions are inexpensive to free, and methods for their use have been firmly established.

Note that this document does not attempt to serve as a primer on XML/XSLT. Rather, it seeks only to introduce them as concepts and explain their roles and benefits. Further technical details are readily available upon request, but beyond the scope of this discussion.

The XML repository

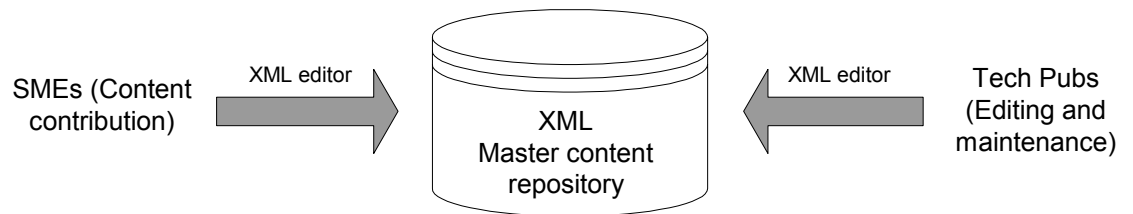
The central XML-based repository is the crux of the proposed new architecture. It could reside as simply an XML file at a shared network location, a set of files, or something more complex such as a database of XML modules. The complexity of such a repository is dictated by the amount of information to store, and the complexity of processes that must act upon it.

For TL1 documentation, a file or file system should be adequate. The transformations to produce the proposed deliverables should not require the advanced features of database storage. Furthermore, database storage immediately adds a higher level of administrative necessity. It is therefore proposed that the effort be initially based on a single XML file, or a simple set of related files.

It should be noted that a file-based system is fully scalable, in the event that a database becomes necessary in the future. All transformations and protocols should be easily adaptable to a different means of XML storage, without the need to rebuild the entire system.

Content contribution (authoring) and editing

Recalling the following portion from the main architecture diagram, all applicable parties are assumed to have access to the central repository:



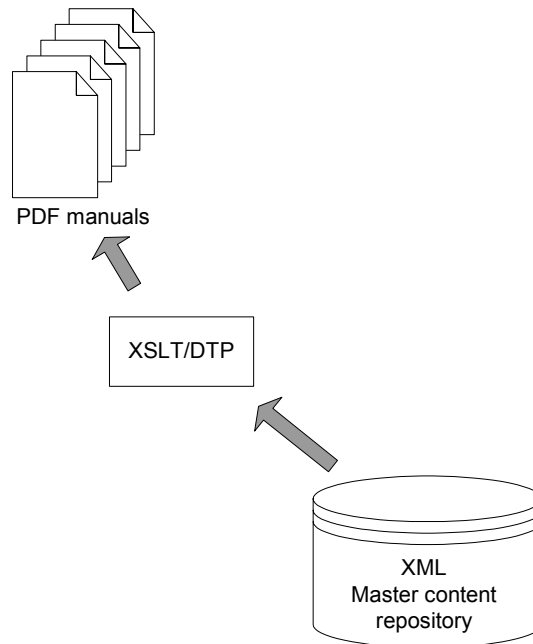
Proposed XML-based architecture for TL1 command documentation

Although protocols, tools, and details must be established, the overall process of authoring is exactly as shown. Using some sort of XML editor or other XML-aware tool, authors and editors access the XML content and contribute directly to the repository in its native format. If the proper tools are used, they will enforce the predefined structural definition of the repository, ensuring perfect consistency and compliance with the transformation technologies that drive the publishing.

The decision whether to author directly in the source files, or to implement some nature of module “check-in/out” is a consideration. Also, the extent of versioning and backups of source content must be considered. These aspects, along with tool choice, are further discussed later in this document.

Publishing PDF manuals

The current format for publishing TL1 command manuals is PDF, via Tech Pubs DTP software. It is assumed, therefore, that PDF manuals will remain an expectation. In the main architecture diagram, the following component appears:



Publishing PDF from XML

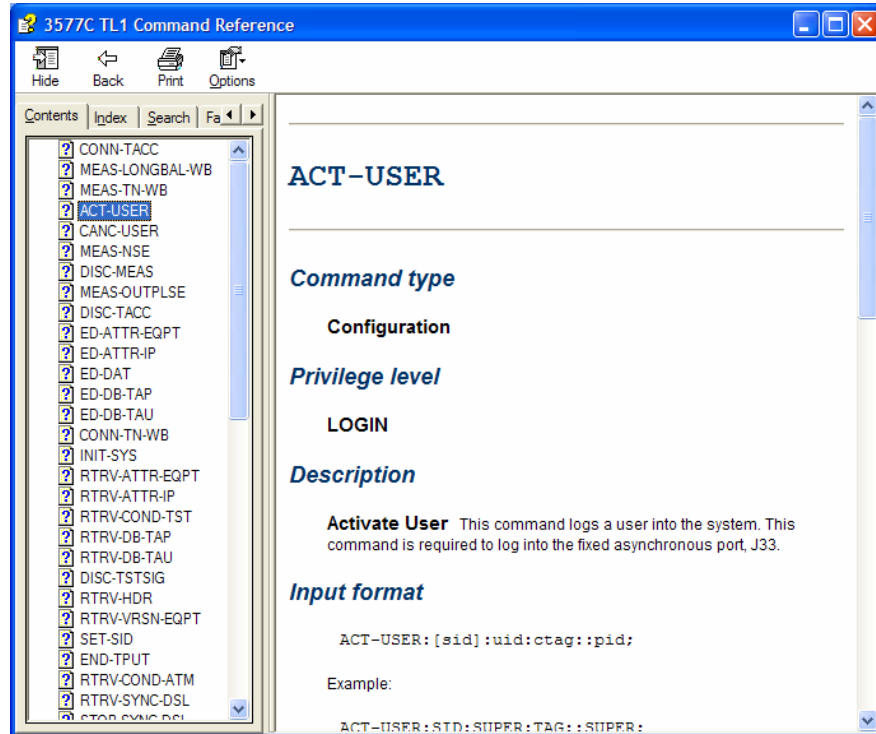
The process in the middle states “XSLT/DTP,” indicating a combined use of transformation and DTP features to mostly auto-generate a PDF manual. The initial plan is to use Tech Pubs current DTP software, which is XML-aware and supports transformation capabilities. In this scenario, instead of Tech Pubs maintaining separate document sets within the DTP, the manuals are generated dynamically through transformations from the XML repository, after which the DTP is simply leveraged for its page layout and PDFing features.

It should be noted that the path to PDF could include many different routes, including a direct XML-to-PDF route using a technology called XSL-FO (XSL Formatting Objects). However, it is recommended to use the current DTP software as an interim step, due to its familiarity, flexibility, and availability within the Tech Pubs group.

Publishing “help file” references

As mentioned earlier, the PDF format is less-than-ideal for publishing most types of command references, due to the challenges in searching and navigation. Newer delivery interfaces have emerged to solve this problem, the most

common being the now-familiar “help file” type of window. The following figure is an example of such a window, portrayed as a hypothetical TL1 command reference:



TL1 command manual displayed through a help viewer

This type of document interface is much better suited for delivery of granular, topic-based content such as a command reference. It provides very fast and convenient features for searching and navigation, in addition to familiar browser conveniences such as flexible window sizing and word wrapping.

Normally, the foundation of these interfaces is HTML, a format readily achievable from XML using XSLT. In fact, the most common, global usage of XSLT is for conversion to HTML for visual rendering. Therefore, an XML-based content repository would allow the delivery of content in this format, whereas it is less feasible now with content stored in a print-centric DTP application.

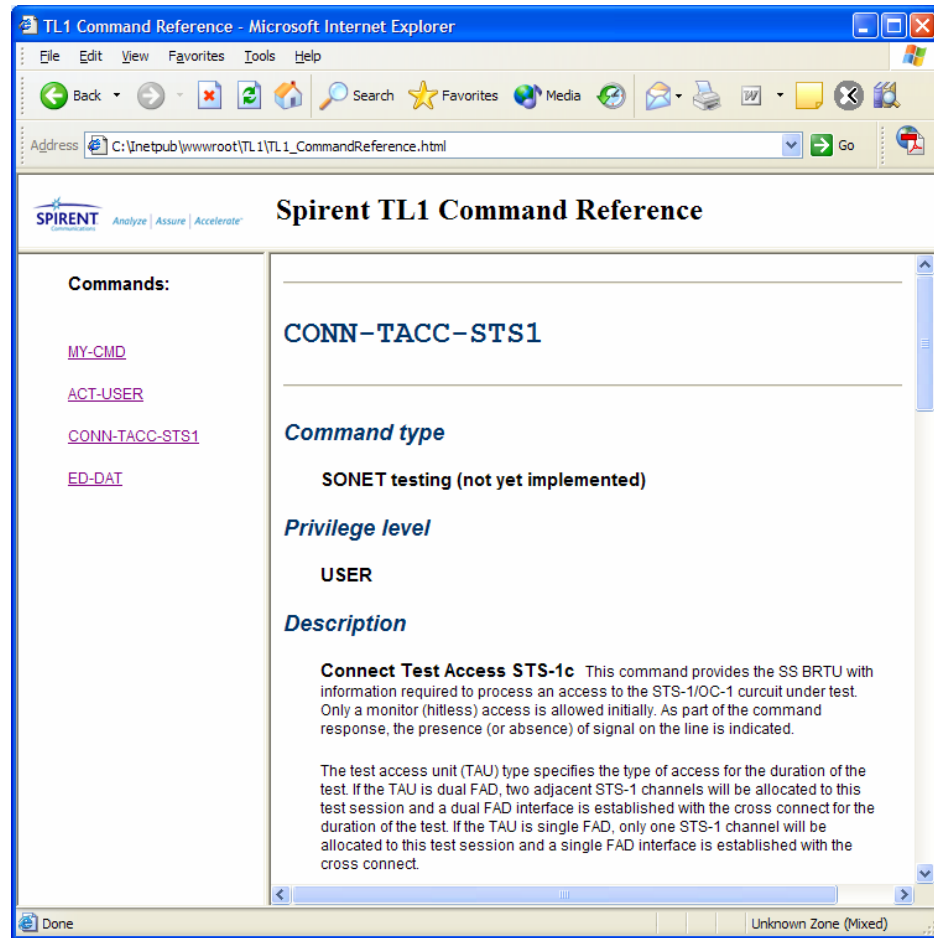
It is acknowledged that concerns over propriety may not allow the delivery of documentation in this format to external customers, due to the very benefits of management and portability it provides. Nonetheless, it is important to note that a significant percentage of TL1 command users are internal to Spirent, perhaps

even a majority. The generation of this format for their convenience would be much appreciated, and the burden of such generation would be greatly reduced by the native XML format.

Publishing a “real-time” reference over the web

In the previous section, it was noted that XSLT is ideal for conversion to HTML, a browser-friendly format. It presented one possibility for creation of HTML by generating static help systems that users could manage as local files.

In addition to this idea, the ability to dynamically generate HTML introduces an even more exciting possibility: a “live” reference that is delivered via the web, either internal or external. In this scenario, users with access would point to a link through their browsers, and some variation of real-time transformation technology would deliver HTML information directly to them over the network. That is, TL1 command information would be delivered dynamically at will, by transforming the requested information directly from the main XML repository. Consider the following prototype web page:



TL1 command reference delivered over the web

This view may look similar to the help file interface in the previous section, and indeed, the same core transformation methods and stylesheets could be used for both. The difference, however, is that this reference would not be static, rather dynamically generated upon request from the main repository. The HTML that users see in the browser is generated on-the-fly through scripted XSLT and does not exist in an HTML file anywhere. Hence, users always view the most updated information available.

This approach brings many advantages:

- No file storage or management is required locally. Only a browser and a network connection are needed.
- Changes to the master repository can be viewed immediately.

- Any nature of information intended for internal consumption only can be stored in the XML and displayed here. Using tags and other metadata, the transformation processes for customer deliverables could ignore internal-only information, while an intranet-based delivery could present all of it with internal information clearly marked.
- A web interface presents a possibility for a web-based check-in/out process for content authoring and editing.

Interest in this type of delivery has been noted within Spirent, and the key would rest in the transformation capabilities of XML. Smaller prototypes of such a system have been built and work very well. A web-based interface could even be deployed to external customers behind a secure extranet, if deemed appropriate.

Architecture summary

The architecture is based on a common XML repository, which is made accessible via common XML tools to any party that requires access. The XML format lends itself to a variety of flexible transformations, including automatic generation of PDF manuals that is currently a laborious human effort. Furthermore, new HTML-based deliveries would become possible through similar transformations, including the possibility of a live web-based delivery of TL1 content.

Note that even without newer HTML-based deliverables, the movement to a common XML repository from which PDF manuals are auto-generated is alone worth the effort. The current amount of inefficiency and redundancy would be greatly reduced by a collaborative authoring system with automated publishing. The ability to deliver via HTML formats is just another significant advantage that would come with the XML.

Proof of concept

Naturally, it is difficult to prove a concept through a document such as this, but it will be at least noted here that prototypes of the proposed architecture have been built and function well. Demos are available from Tech Pubs on request.

Steps towards implementation

The following sections outline the suggested approach towards implementation of the proposed architecture. For the purposes of this document, these steps are necessarily broad and meant as a preliminary guideline only.

Step 1 – Form a collaborative team

The effort to fully implement and design this architecture must involve all affected parties, including, but not limited to:

- Tech Pubs
- Current and potential content contributors
- Content users
- Interested management

Key individuals should be identified from these entities for collaboration throughout the project, such that the final outcome is assured to be acceptable to all.

Step 2 – Determine exactly what a TL1 command reference should consist of

Using the current command manuals as a starting point, the collaborative team should finalize what the structure of command documentation should look like, and identify all types of content it will contain. It is possible that the current format is largely acceptable, and this step will be little more than an exercise to analyze the current structure for conversion to XML. It is more likely, though, that current deficiencies will be identified and the team can take advantage of this opportunity to optimize the content structure of our TL1 documentation.

In any case, it is critically important to nail down the content structure beforehand, because the XML structure definition will be based on it. Once the transformation processes are built around the XML structure definition, the definition will be more difficult to change.

Step 3 – Design the XML DTD/schema for the repository

The XML structure definition will be the roadmap to the content, and thus the foundation for all transformation instructions. It should be finished, as best as possible, and collaboratively approved before authoring and publishing processes are built around it. This step should also include the determination of how the repository should be set up, such as files and locations.

Step 4 – Design the publishing processes

The most pressing need from the repository will be publishable deliverables. A primary goal is to provide collaborative authoring in the central repository, but this step can wait until the repository and publishing architectures are complete.

Using the determination of output types and content structures, the transformation processes should be built around the stable XML structure definition. This step involves not only transformation technologies, but also

collaboration on the content and appearance of deliverables, such as colors, fonts, and other visual parameters.

Step 5 – Migrate existing content to XML

Once the transformation and publishing architectures are nearing completion, the team can be reasonably sure that the XML structure definition has matured. At this point, it would be logical to begin the migration of legacy content into the XML repository. It is also logical to ensure that all publishing works as intended before the significant effort of migration begins.

While simple in concept, this step will involve a considerable investment. The layout of the legacy content is not very good and will take a significant amount of manual work to convert into a proper, hierarchical XML format. The nature of the content is perfectly suited for XML, but the current layout is not well-suited for easy conversion.

Furthermore, it has been noted that many errors exist in the content itself, which indicates the need for a solid engineering review. A period of migration is ideal for this effort, but should be expected that it may add additional time. This would be time well-spent in any case, because accurate content is vitally important to the quality of deliverables, and should be pursued regardless of the publishing architecture.

Note that the proposed architecture assumes that the entire TL1 command library across products will ultimately be stored in the same repository. This does not mean, however, that all must be migrated before the repository can be used. It has been suggested that only the commands from one particular product be migrated to start, allowing a further proof-of-concept and a more incremental transition.

Step 6 – Determine and establish authoring protocols

XML can be authored and edited by any XML-aware tool. The most convenient and user-friendly tool should be chosen for the migration effort, due to the

burden of work. However, once “normal” authoring in the live system must begin, a huge variety of tools and protocols are available. The collaborative team must determine what tools to recommend to users, and how the repository should be accessed and protected.

This document is necessarily vague on this point, because many possibilities exist, and it will be up to the collaborative team to determine the best approach. While working, the team should consider the following:

- A suitable XML authoring tool must support strict validation. The transformation routines may be severely hampered by content that does not adhere to the prescribed structure definition.
- A suitable XML tool must be user-friendly in the area of “XML document authoring.” A wealth of XML tools exist, but many are focused more on data-centric XML functions and are less appropriate for document-style XML.
- The repository itself must have some form of back-end validation.
- A protocol/process for accessing the repository must be established. By the time this architecture reaches full capacity, it will be no longer feasible to have many contributors working directly in the source XML file(s). Some type of check-in/out process will likely become necessary.

Considerable flexibility exists with accomplishing this step. If changes or modifications become necessary, this should have little to no effect on the transformation/publishing architecture, provided that all efforts ultimately provide XML that adheres to the established structural definition.

Step 7 – Deploy and train

This step largely speaks for itself, and the ease of accomplishment will depend entirely on how well the overall project is conducted with concerns for usability.

Roles

At the time of this writing, it is assumed that Tech Pubs will take responsibility for the entire construction of the new infrastructure. External personnel, such as the aforementioned collaborative team, will be heavily relied upon for design decisions and review, but ultimately Tech Pubs will build it.

Challenges

While this proposal suggests significant rewards and benefits, the implementation would not be without significant challenges. The following sections address the major challenges identified thus far.

Migration of legacy content

The migration effort is a challenge due to the sheer volume of work involved. It is not necessarily a technical challenge, rather a manpower challenge. While Tech Pubs has plenty of experience with this type of migration, the expertise cannot negate the manual cleanup that will be caused by the rudimentary layout of the current content. It is extremely difficult to estimate such a number, but a preliminary guess would be that the effort would require workhours in the low hundreds.

During an unstructured-to-structured content conversion, scripting can be used wherever automated routines are possible. Automated routines are only effective when the existing content has some sort of implied hierarchy through formatting conventions, and the current TL1 documentation set lacks much of this. Nonetheless, automated processes can perform some of the work and will be used.

Despite the migration costs, the effort should be deemed worthwhile overall. Undoubtedly, many hundreds of workhours have been expended solely on the inefficiencies of the current process, which has still produced a less favorable

product. This trend will certainly continue until the investment in change is made. Furthermore, once content is in a structured, metadata-rich format, any future conversions (if necessary) will lend themselves well to automation. Incidentally, Spirent Tech Pubs has been following an important industry trend towards structuring all technical content, and this conversion may have occurred anyway. It would be best to combine this effort with a migration straight to XML and this new architecture, which actually adds very little extra work to the already pending conversion process itself.

Development of a technology-driven infrastructure

The storage of XML data and automated publishing thereof will require a higher level of technology than Spirent Tech Pubs is accustomed to. In brief, the proposed technologies are as follows:

- **XML repository**—A file or file system on a network server.
- **Authoring and other contribution**—Validation-supporting XML editors, with preferably a check in/out system on the front-end of the repository.
- **Publishing PDF manuals**—Transformation technologies working in conjunction with the current PDF-ready DTP software to auto-generate manuals from the repository.
- **Publishing help file-style references**—Basic transformation technologies to auto-generate static HTML pages, with perhaps a modest Java-driven supervisory front end for coordination.
- **Publishing a web-based reference**—HTML auto-generated by basic transformation technologies, driven by ASP on a Windows server.

The majority of these items has been successfully prototyped and works very well. The challenges rest mostly in the expansion and refinement for production, with a heavy emphasis on usability. In general, it should be a moderate effort to get the basic infrastructure working, with refinements possible over time. Overall, this is deemed less of a challenge than the legacy content migration effort, at least in terms of sheer workhours.

Gaining acceptance

Naturally, the primary factor for successful implementation will be getting people to use it. It is hoped that with a strong focus on usability, acceptance will be reasonably smooth. If the proposal lives up to its promises and makes everyone's job easier, not harder, acceptance should be automatic. Thus far, all demos and discussions have met with very positive results, indicating that most, if not all, interested parties realize the importance of a change.

On a related note, it will take a small amount of time to train affected parties on how it all works. But again, if the overall result is less work and better output for all, this should not present a major hurdle.

Risks

No ambitious proposal such as this comes without risk. The following table outlines some identified risks identified thus far.

Risk	Response
<p>The underlying technology is too complex to build or maintain by Tech Pubs</p>	<p>As mentioned earlier, most of the proposed technology has been successfully prototyped by Tech Pubs, indicating that it is currently within reach. Certainly, though, the loss of certain key personnel could result in the short-term inability of the department to complete and/or maintain the system.</p> <p>The key to success and longevity, therefore, will be simplicity and proper documentation. Although not normally the domain of a technical writer, many writers do possess advanced XML skills. Furthermore, Spirent deals heavily with XML and transformations in other departments, so a wealth of expertise resides on staff. If the overall system is kept modest, it should be reasonably straightforward for any employee with XML expertise to step in and assist.</p>

<p>The system simply doesn't deliver as advertised, and the new deliverables weren't worth it</p>	<p>The flow and output of the proposed architecture has been carefully considered using industry standards as a model. All involved technologies have ample precedent throughout the techcomm industry. And, the proposed outputs have been carefully compared with current TL1 documentation output to identify the important benefits. The case is very strong to implement collaborative authoring on this material and to deliver the content through better methods. Working prototypes, which took only a few days to build, show that enormous potential exists for a modest amount of technical investment.</p>
<p>As a "home-grown" system, it gradually becomes entrenched and difficult to migrate</p>	<p>Currently, no feasible "over-the-counter" solution to the TL1 issues exist, so this proposal necessarily includes custom components. Nonetheless, note the following relevant points:</p> <ul style="list-style-type: none"> • The structure definitions will be adherent to the Darwin Information Typing Architecture standard (DITA), facilitating a seamless move to an OTC tool should one become available. • The nature of XML is portability and transformability. And, it is the foundation of nearly all advanced content management systems. The conversion to XML would likely be the necessary step to an OTC solution, so this proposal largely moves us in this direction anyway.

Summary

Modern techcomm methods and technology have presented an opportunity to upgrade the process and delivery of TL1 command documentation at Spirent. Following a period of investment, Spirent can realize an improvement in the quality of documentation output, while simultaneously increasing the efficiency of its authoring and production. Headed by Tech Pubs, a move in this direction is important for TL1 documentation, and also lays the groundwork for similar

improvements in other areas. It is time to use the tools at our disposal to free ourselves from the burdens of now-obsolete legacy processes.