International Journal of Computer Applications (IJCA)
Volume 5, Issue 2, July-Dec 2024, pp. 14-19, Article ID: IJCA_05_02_002
Available online at https://iaeme.com/Home/issue/IJCA?Volume=5&Issue=2
ISSN Online: 2341-7801, DOI: https://doi.org/10.5281/zenodo.14257993
Impact Factor (2024): 7.22 (Based on Google Scholar Citation)

© **IAEME** Publication SCOPE DATABASE INDEXED

OPEN ACCESS

DESIGN AND IMPLEMENTATION OF A SECURE PRINTER SERVER FOR RESTRICTED WAREHOUSE LABELING OPERATIONS

Gautham Ram Rajendiran
USA

ABSTRACT

This research paper presents a detailed exploration of implementing a Printer Server to facilitate printer and labeling services in restricted warehouse environments []. The Printer Server serves as a centralized bridge connecting a warehouse portal with local and remote printers, enabling streamlined label printing within controlled network constraints. This system supports diverse label formats, such as ZPL and PDF, across both local and network-connected printers. It includes solutions for overcoming mixed-content browser restrictions [2] and introduces mechanisms for effective network management within the warehouse. The solution overview highlights the architecture, API details, handling mixed content issues, and specific implementation challenges to achieve reliable printing operations in restricted environments.

Keywords: Printer Server, Warehouse Labeling, Secure Printing, Network-Restricted Environments, Remote Printer Management, Spring Boot Framework

Cite this Article: Gautham Ram Rajendiran, Design and Implementation of A Secure Printer Server for Restricted Warehouse Labeling Operations, International Journal of Computer Applications (IJCA), 5(2), 2024, pp. 14–19.
https://iaeme.com/Home/issue/IJCA?Volume=5&Issue=2

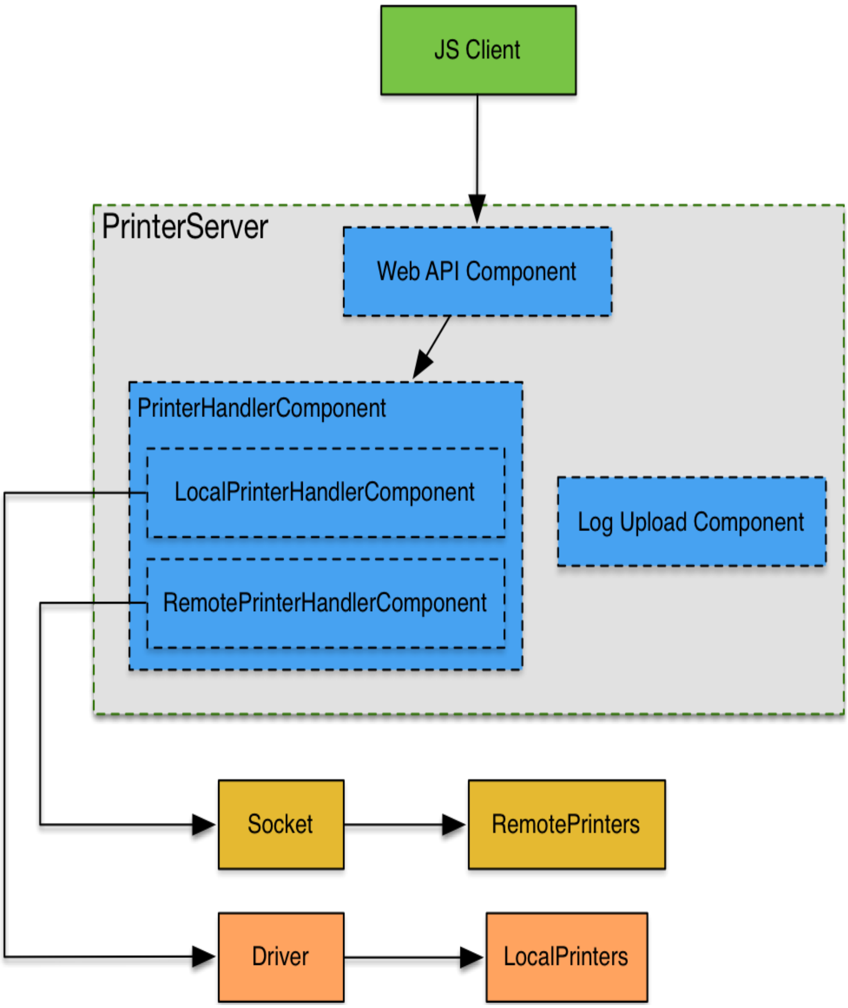
INTRODUCTION

In warehouse environments, efficient labeling and printing capabilities are critical for inventory management and streamlined operations. However, these operations often require restrictive network configurations to maintain security and integrity, which poses unique challenges in managing and controlling document printing. This paper explores the design and implementation of a Printer Server that enables secure, reliable printing within these controlled environments. This Printer Server bridges the gap between the warehouse's web portal and physical printers, receiving HTTP [3] requests from the portal and redirecting them to designated printers for label printing. The system ensures that all requests are effectively processed while maintaining secure network boundaries and adhering to warehouse protocols. In addition, we detail the functional scope, network requirements, and API specifications that allow the Printer Server to operate seamlessly in these restricted settings.

System Architecture and Features

The Printer Server plays a central role in the printing ecosystem, acting as an intermediary between the warehouse portal and the physical printers. The system architecture is designed to accommodate both local and remote printers, with specific configurations tailored to each setup. The Printer Server is deployed on a dedicated desktop within the warehouse, receiving and processing label-printing requests from the warehouse's web-based portal. For local printers, the Printer Server connects via USB [4] or local network configurations, while remote printers are accessed over LAN, binding the server and printers with static IPs to ensure uninterrupted communication. Given the secure nature of the warehouse environment, this system omits document encryption, as no UCI (Unclassified Controlled Information) [5] data is involved.

The Printer Server's capabilities include support for multiple printer types, such as ZPL and HP models, allowing for versatile label formatting options. Additionally, the system supports both local and remote printer invocations, facilitating flexible printing configurations depending on the warehouse's specific requirements. Each printer and the Printer Server itself are assigned static IPs to maintain consistent access and network reliability. API requests are sent to the Printer Server from the web portal, enabling users to initiate, monitor, and control the printing process through standardized HTTP requests. This centralized control is integral to maintaining a streamlined and reliable printing process within the warehouse.



SOLUTION DETAILS

The Printer Server's implementation leverages the Spring Boot framework, enabling a robust, scalable HTTP server designed to handle label-printing requests from the warehouse's web-based portal. This setup uses JSON as the data format for requests and responses, facilitating straightforward integration with the web interface while ensuring the simplicity and clarity of data transmission. To address cross-origin requests or "mixed content" issues common in restricted environments, the Printer Server is configured with Access-Control-Allow-Origin headers, ensuring that all requests within the local network (e.g., 192.168.1.x) are processed without interference.

The Printer Server's architecture supports both local and remote printer management. For local printers, which are typically connected via USB or configured directly on the desktop where the Printer Server is running, the system employs the javax.print library. This library enables direct communication with the printers and manages queued print jobs through the Windows Print spooler, ensuring each print job is handled in sequence and minimizing potential conflicts. Remote printers, in contrast, are connected via LAN and are accessed through a Socket connection, which allows direct communication with printers bound to specific IP addresses. To prevent conflicts, the Printer Server includes a locking mechanism that serializes print requests, ensuring that only one job is processed by each printer at any given time.

Browser security measures that prevent mixed content are a common challenge in restricted environments. In response, we have configured the Printer Server to accommodate specific settings in Chrome and Firefox to allow insecure content. Users running Chrome on macOS, for instance, can enable insecure content by selecting the "shield" icon within the browser or by configuring the application with the --allow-running-insecure-content argument. Similarly, in Firefox, users can adjust settings via the about:config page to disable mixed content blocking, ensuring consistent access to the Printer Server despite these restrictions.

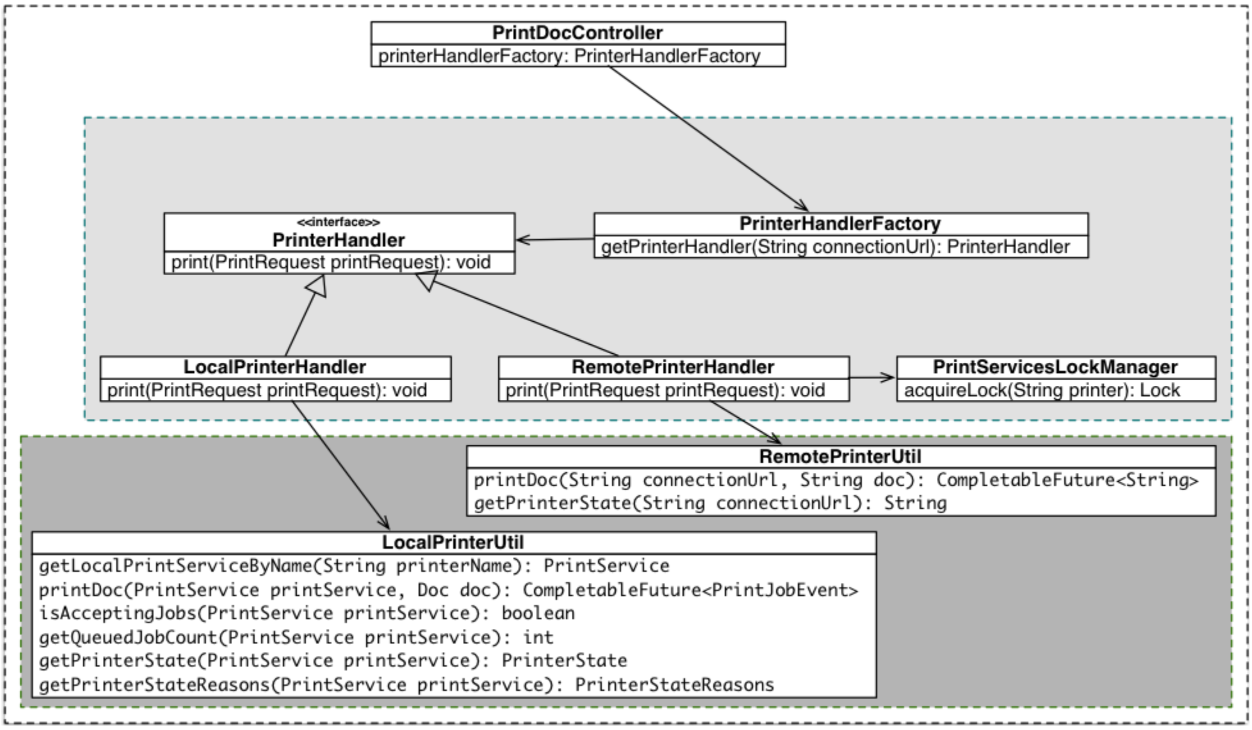
Components of Printer Server

Name	Input	Output	Mark
Print	String connectionUrl; String document; int numberOfPrintCopies; String documentFormat; [ZPL, PDF]	void	document is a String encoded in Base64. Error code in Exception: PRINT_JOB_FAILED PRINTER_NOT_FOUND INTERNAL_ERROR PRINTER_NOT_READY
Ping	void	String version;	
Shutdown	void	void	

The Printer Server consists of multiple components designed to streamline request processing and enable efficient printer management. At the core of this setup is the PrintDocController, which acts as the main controller, handling all incoming requests and directing them to appropriate printer handlers based on whether the request is intended for a local or remote printer. This routing ensures that each request is processed accurately and in alignment with the specific requirements of each printer type.

To address the distinct operational needs of local and remote printers, the Printer Server includes two handler classes: LocalPrinterHandler and RemotePrinterHandler. These handlers contain business logic tailored to the requirements of their respective printers, ensuring that each print job adheres to the constraints and configurations needed for successful execution. Additionally, the system includes LocalPrinterUtil and RemotePrinterUtil utilities, which directly interact with each printer model, translating handler requests into appropriate commands based on each printer's format requirements.

The Printer Server's design enables multiple instances to access a single printer, introducing a critical need for request control. For local printers, the system leverages the Windows Print spooler, which naturally manages queued print jobs. For remote printers, the Printer Server enforces a locking mechanism that processes requests sequentially, preventing simultaneous access to the same printer and ensuring each print job is executed without conflict.


PrintDocController
printerHandlerFactory: PrinterHandlerFactory
<<interface>>
PrinterHandler
print(PrintRequest printRequest): void
PrinterHandlerFactory
getPrinterHandler(String connectionUrl): PrinterHandler
LocalPrinterHandler
print(PrintRequest printRequest): void
RemotePrinterHandler
print(PrintRequest printRequest): void
PrintServicesLockManager
acquireLock(String printer): Lock
RemotePrinterUtil
printDoc(String connectionUrl, String doc): CompletableFuture<String>
getPrinterState(String connectionUrl): String
LocalPrinterUtil
getLocalPrintServiceByName(String printerName): PrintService
printDoc(PrintService printService, Doc doc): CompletableFuture<PrintJobEvent>
isAcceptingJobs(PrintService printService): boolean
getQueuedJobCount(PrintService printService): int
getPrinterState(PrintService printService): PrinterState
getPrinterStateReasons(PrintService printService): PrinterStateReasons

CHALLENGES AND SOLUTIONS

One challenge encountered during development was the ZM400 printer's limitation in handling multi-copy printing jobs within a single print request due to constraints in the javax.print library. To address this, the Printer Server utilizes the Windows Print spooler to manage print job queues, sending each copy request individually rather than as a bulk operation. This solution allows the Printer Server to maintain compatibility with the ZM400's restrictions while ensuring multi-copy functionality.

Another challenge involves mixed content issues stemming from browser security protocols, which prevent the display of HTTP content when accessed over secure HTTPS pages. To overcome these barriers, we provided configuration instructions for Chrome, Firefox, and Windows to allow "insecure" content [6].

For instance, users can configure Chrome to bypass these restrictions by adding the --allow-running-insecure-content parameter. Similarly, Firefox users can adjust settings in about: config to control mixed content blocking. These configurations are essential in maintaining a seamless printing experience despite the security limitations inherent in restricted environments.

Printer access synchronization is another critical challenge in managing remote printers accessed via Socket connections. The Printer Server includes a lock mechanism [7] that ensures print requests to remote printers are handled one at a time, avoiding potential conflicts. This mechanism is crucial in restricted environments where concurrent access may lead to job failures or misprints, ensuring a controlled and organized printing process.

SAMPLE REQUESTS AND IMPLEMENTATION

The Printer Server offers several APIs to manage printing tasks, each accessible via HTTP requests. For instance, the Ping API enables users to check the Printer Server’s operational status by sending a simple POST request to the /ping endpoint, which returns the server’s version details. The Print API allows users to submit print jobs by posting a JSON payload containing parameters such as the document to be printed, document format, and the number of copies requested. This request can accommodate both PDF and ZPL [8] formats, enabling flexible label printing options. Finally, the Shutdown API allows users to remotely deactivate the Printer Server, ensuring operational control and system integrity in the restricted warehouse environment.

CONCLUSION

The Printer Server provides a robust and adaptable solution for printing and labeling services within restricted warehouse environments. By centralizing control over printing tasks and enabling secure communication with both local and remote printers, the Printer Server enhances operational efficiency and supports critical labeling functions essential to warehouse management. Through its careful integration with network protocols, handling of mixed content issues, and printer access synchronization, this system addresses unique challenges inherent to restricted environments. Future work will explore additional capabilities such as document encryption, expanded multi-copy support, and advanced printer health monitoring to further enhance the Printer Server’s utility in secure environments.

REFERENCES

[1] D. S. Alexander, et al., "A secure active network environment architecture: Realization in SwitchWare," IEEE Network, vol. 12, no. 3, pp. 37-45, 1998.

[2] A. Kulshrestha, "An empirical study of HTML5 websockets and their cross browser behavior for mixed content and untrusted certificates," International Journal of Computer Applications, vol. 82, no. 6, 2013.

[3] D. Gourley and B. Totty, HTTP: the definitive guide, O'Reilly Media, Inc., 2002.

[4] J. Axelson, USB Complete: The Developer's Guide, Lakeview Research LLC, 2015.

[5] R. Ross, et al., Protecting Controlled Unclassified Information in Nonfederal Systems and Organizations, NIST Special Publication (SP) 800-171 Rev. 2 (Draft), National Institute of Standards and Technology, 2019.

[6] P. S. Satish and R. K. Chavan, "Web browser security: different attacks detection and prevention techniques," International Journal of Computer Applications, vol. 170, no. 9, pp. 35-41, 2017.

[7] R. Guerraoui, et al., "Lock–unlock: Is that all? A pragmatic analysis of locking in software systems," ACM Transactions on Computer Systems (TOCS), vol. 36, no. 1, pp. 1-149, 2019.

[8] L. Snyder, "The design and development of ZPL," in Proceedings of the third ACM SIGPLAN conference on History of programming languages, 2007.

Cite this Article: Gautham Ram Rajendiran, Design and Implementation of A Secure Printer Server for Restricted Warehouse Labeling Operations, International Journal of Computer Applications (IJCA), 5(2), 2024, pp. 14–19

Abstract Link: https://iaeme.com/Home/article_id/IJCA_05_02_002

Article Link:
https://iaeme.com/MasterAdmin/Journal_uploads/IJCA/VOLUME_5_ISSUE_2/IJCA_05_02_002.pdf

✉ **editor@iaeme.com**