4. 教育用計算機環境の TCO 削減にむけて

1. Ridoc IO Gate

（株）リコー
ソリューションマーケティングセンター
岸保 直人
ganbo@sdg.mdd.ricoh.co.jp

概要とねらい

教育用計算機システムのプリンタにおいて，いかに学生の無駄な印刷を抑え印刷トラブルを減らせるかは TCO（Total Cost of Ownership）を削減する上で大変重要な課題である．学生の印刷コストに対する認識は低く，無駄な印刷はやめるよう呼びかけてもあまり効果はない．また，プリンタパネルの誤操作に職員が振り回される例も後を絶たない．こういった状況を改善するのがプリント管理システム「Ridoc IO Gate」（リドック・アイオー・ゲート）である．本編では，Ridoc IO Gate が提供する 3 つの機能，（1）印刷上限管理，（2）オンデマンド印刷，（3）課金印刷について，その仕様や実装に関する技術的側面について紹介する．

プリント管理の必要性

教育用計算機システムのプリンタは授業の一環として学生に開放されることが多いが，近年は Web やマニュアルなど電子的な情報ソースの増加に伴って印刷量が増大し，用紙にかかるコスト負担が増えつづけている．

プリンタの使用状況を調べてみると，学生の印刷コストに対する意識の低さを嘆く大学関係者の声は多い．プリンタは無料で利用できるため，とりあえずといった安易な印刷，あきらかに私用で大量印刷しているという状況もある．また，学生の知識不足によるプリンタパネルの誤操作や，故意の設定変更などにより印刷できなくなるといったトラブルに職員が振り回され，本来の業務に支障をきたす例も後を絶たない．

こうした状況を改善する方策の 1 つとしてプリント管理システムを導入するケースが最近増えてきている．

プリント管理システム「Ridoc IO Gate」

プリント管理システムと一言でいっても目的や投資コストなどによってその形態はさまざまである．リコーの場合，プリント管理システムが提供するソリューションを 4 階層に分類している．

Stage1. 印刷枚数管理
ユーザ（学生）の印刷枚数を集計．グラフ掲示などをすることによりプリンタ利用方法の改善を促す．

Stage2. 印刷上限管理
ユーザの印刷枚数を集計し，ユーザ個々に許可された印刷枚数の上限を超えた場合に印刷を不可にする．

Stage3. オンデマンド印刷
クライアントから印刷指示を出すと，サーバが印刷データをスプール領域でいったん保存．プリンタへ出力するには，プリンタ横に設置してある操作端末にて改めて出力指示．印刷上限管理（Stage2）に加えて印刷先の間違いや印刷物の取り忘れを防ぐ．

Stage4. 課金印刷
オンデマンド印刷（Stage3）の操作端末にコインラック，プリペイド，IC カードといった課金装置を接続し出力枚数に応じて課金．受益者負担の考えで印刷コストを削減する．

プリンタの TCO 削減には，Stage1 の印刷枚数管理では学生の自主性に頼るためか効果が薄く，システムが強制的に制限をかける Stage2 以上の仕組みが必要となることも多い．

リコーでは各 Stage に適したパッケージ商品を提供しているが，その中で Stage2 から Stage4 のソリューションを提供するのが「Ridoc IO Gate」である．Ridoc IO Gate の基本構成はサーバ上で稼働するプリント管理ソフトウェアと専用プリンタ．メンテナンス性を考慮してクライアントには特別なソフトや設定は不要な作りになっている．サーバのプラットフォームは UNIX（Sun Solaris），Linux，Windows に対応しており，さまざまな教育計算機

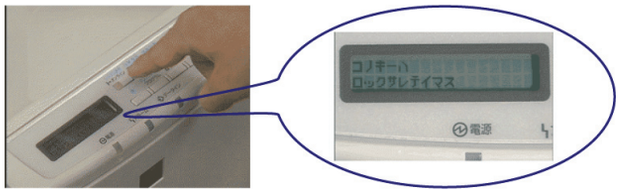

図 -1　操作パネルのロック

環境に柔軟に対応できる．この Ridoc IO Gate の仕様や実装に関する技術的側面についてこれから紹介する．

専用プリンタ

不正アクセス防止

Ridoc IO Gate の印刷データにはプロテクトデータが付加されるが，プリンタ側でその値をチェックし不正な場合は印刷せずに破棄する．これにより不正アクセスによる印刷を防止する．

パネルロック

印刷中のジョブを中断させる「ジョブリセット」キーを除き，プリンタ操作パネルのすべてのキーをロックする（**図 -1**）．これにより，IP アドレス変更やオフラインといった，プリンタ設定の変更によるトラブルを未然に防ぐ．

その他にも，IP アドレスのアクセスコントロールマスクにより外部のネットワークや登録外のクライアントからのアクセスを制限する機能や，印刷データがポストスクリプトであれば不測のデータによる大量の化け文字印字を防止する機能なども搭載している．

印刷上限管理

クライアントで作成された印刷データは，OS 標準のプリントシステムが受け取り，プリントシステムから呼び出される Ridoc IO Gate のプログラムが次の手順で印刷上限管理の処理をする（**図 -2**）．

①クライアントから印刷サーバへデータを送信する
②印刷サーバがデータを受信すると，印刷データをプリンタへ送信する許可を印刷ログを集中管理しているサーバ（ログ管理サーバ）に問い合わせる．ログ管理サーバから印刷不可の返答がきた場合，印刷データを破棄する
③印刷データに次の変更を加え，プリンタに送信する

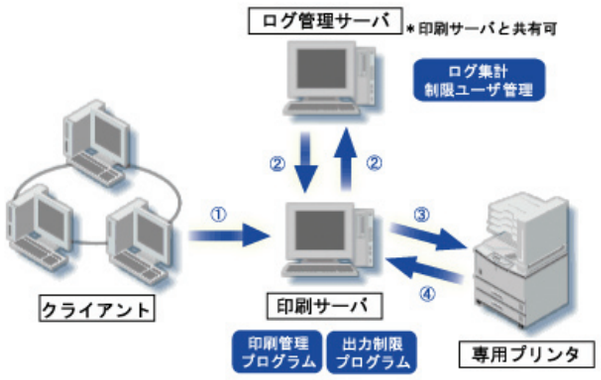

図 -2　印刷上限管理の流れ

- 正規ルートで出力されたことを証明するプロテクトデータを付加
- フッターにユーザ名と印刷日時を付加　※オプション
- 1 印刷ジョブあたりの印刷制限枚数を指定する制御データを付加　※オプション

④プリンタから実際に印刷された枚数を取得し，ログ管理サーバに印刷ログとして送信する

ページ記述言語にはポストスクリプトと RPCS（リコーオリジナル）をサポートしているが，ポストスクリプトの場合を例にとり，上記③で印刷データに変更を加える仕組みを示す．

プロテクトデータ

プロテクトデータは印刷データの属性情報を元に作成するため，同じ印刷データでも印刷するユーザや印刷日時によって異なる値になる．

実際のプロテクトデータはプリンタジョブコントロール言語である PJL[☆1] で次のように記述する．

```
@PJL SET RIOGJOBPASSWORD=9007
```

PJL で環境変数に値を設定するフォーマットは，@ PJL SET < 環境変数 > = < 値 > になっており，この例では，プロテクトデータの環境変数 RIOGJOBPASSWORD に値「9007」を設定している．

印刷データを受け取ったプリンタはサーバと同じアルゴリズムでプロテクトデータを計算し，その結果が RIOGJOBPASSWORD の値と同じになるかチェックするわけである．

フッター

ユーザ名，出力日時を印刷物の全ページ左下に付加することで，取り違えや取り忘れを抑止する．

Ridoc IO Gate のフッター印刷の仕組みはシンプルであ

☆1 PJL は，Hewlett-Packard 社が開発したプリンタジョブコントロール言語で，プリンタ言語の上位に位置する．リコーではこの言語に準拠したリコー仕様の PJL を独自に機能拡張し，プリンタ機能を十分に活用できるようにしている．実際の印刷データのフォーマットは，PJL の記述領域がプリンタ言語の記述領域を前後にはさむようになっている．

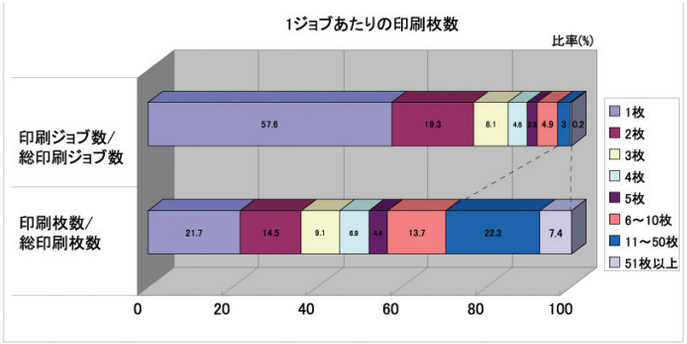

図 -3　プリンタ利用実態の事例

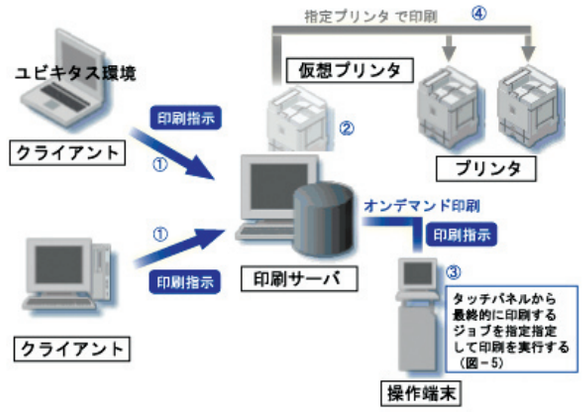

図 -4　オンデマンド印刷の流れ

る．ポストスクリプトの改ページ命令に相当するオペレータ（showpage など）を見つけ出し，その前に次のような命令を付加する．

```
gsave initgraphics
/GothicBBB-Medium-90ms-RKSJ-H findfont 8
scalefont setfont 20 20 moveto
（ユーザー名: ricoh　印刷日時: 2003/3/10 15:20:30）
show grestore
```

上記の例は，「ユーザ名：ricoh　印刷日時：2003/3/10 15:20:30」という文字列を紙の左下位置を原点に x,y 軸とも 20 ポイント移動したところからフォント「GothicBBB-Medium-90ms-RKSJ-H」で印字することになる．なお，印字開始位置やフォントなどはシステム管理者が変更可能だ．

1 ジョブ内の印刷枚数制限

月間や年間の上限印刷枚数を設定するだけでなく，1 ジョブ内での印刷制限枚数を設定することでケアレスミスや計画性のない大量印刷を抑制することができる．

制限枚数を何枚にすべきかは運用環境に依存するため一概に言えないが，ある教育計算機環境における 1 ジョブあたりの印刷枚数統計を参考までに示す（図 -3）．

この例の場合，制限枚数を 10 枚に設定すれば，総印刷ジョブの 97% は制限にかからず印刷できる上で，仮に制限枚数を超えるジョブを印刷せずに破棄すれば紙の消費量を 70% に抑えられることが分かる．また，このグラフのデータだけでは情報不足だが，仮に制限枚数を超えるジョブを制限枚数を超えたところで印刷ストップすれば，紙の消費量は 82% に抑えられる計算結果になった．

1 ジョブ内の印刷制限枚数を実装するには各種方法が考えられるが，Ridoc IO Gate ではサーバ，プリンタ，印刷データに仕組みを入れる 3 通りの方法を提供している．システム構成による対応可／不可，導入コスト，ファーストプリント性能，複数部数印刷時の動きなどが各方式で異なるためだ．

［サーバ方式］

印刷中のジョブがプリンタに現在何枚目のデータを送信中であるのかを Windows であればスプーラやポートモニタから教えてもらい制限枚数に達したら印刷処理を中断する．

［プリンタ方式］

印刷データのヘッダーに PJL で印刷制限枚数を次のように設定する．プリンタがページ展開処理のタイミングで枚数をチェックし，制限枚数以上の印刷ジョブなら印刷せずに破棄する．

```
@PJL SET RIOGLIMITPAGES=50
←意味：印刷制限は50枚に設定
```

［印刷データ方式］

ポストスクリプトの改ページ命令に相当するオペレータ（showpage など）を印刷データの先頭で次のように再定義する．showpage するたびに，変数 rioglimitpage に設定されている制限枚数の値（50）がカウントダウンしていき，0 になった時点で強制的に印刷を終了する仕組みだ．

```
/rioglimitpage 50 def
/orgshowpage /showpage load def
/riogshowpage {
rioglimitpage 0 eq {stop} {orgshowpage} ifelse
/rioglimitpage rioglimitpage 1 sub def
} def
/showpage /riogshowpage load def
```

オンデマンド印刷

オンデマンド印刷は，実際にプリンタから印刷する前に，印刷するかどうかの最終確認をしたり，印刷に使用する物理的なプリンタを指定できるシステムだ．（図 -4, 図 -5）

①クライアントからの出力先に，Ridoc IO Gate 対応プリンタの替わりに仮想プリンタを選ぶ．仮想プリンタとは，同種類の複数台数のプリンタをまとめたもので，クライアント側からは 1 つのプリンタとして扱われる．

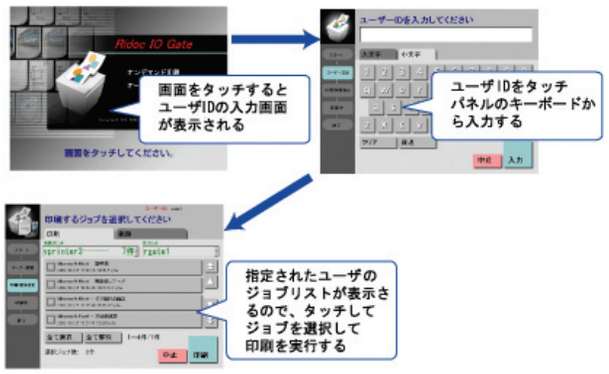

図 -5　専用端末画面例

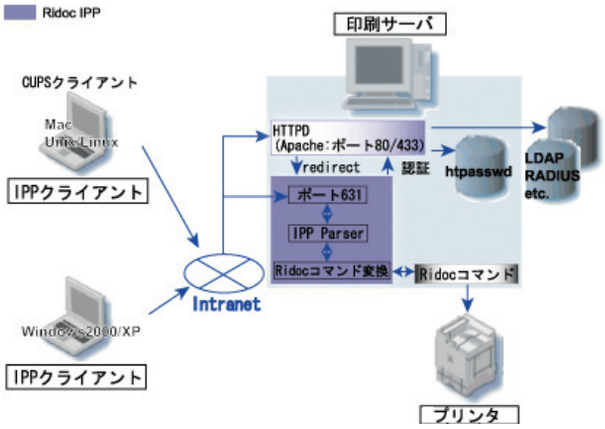

図 -6　Ridoc IPP の流れ

内容	IPPオペレーション（オペレーション番号）	Ridoc IO Gate 印刷コマンド
印刷ｺﾏﾝﾄﾞ	Print-Job(0x0002)	/opt/rgate/pub/bin/rgprintjob
ｼﾞｮﾌﾞ状況確認ｺﾏﾝﾄﾞ	Get-Job-Attributes(0x0009)	/opt/rgate/pub/bin/rggetjobattrs
ｽﾌﾟｰﾙ状況確認ｺﾏﾝﾄﾞ	Get-Jobs(0x000A)	/opt/rgate/pub/bin/rggetjobs
ｷｬﾝｾﾙｺﾏﾝﾄﾞ	Cancel-Job(0x0008)	/opt/rgate/pub/bin/rgcanceljob
ﾌﾟﾘﾝﾀ状況確認ｺﾏﾝﾄﾞ	Get-Printer-Attributes(0x000B)	/opt/rgate/pub/bin/rggetprnattrs

表 -1　Ridoc IO Gate 印刷コマンドへのマッピング

②仮想プリンタに送信された印刷データはサーバの専用スプール領域に保存される．

③ユーザは操作端末のタッチパネルを想定したユーザインタフェースから ID とパスワードを入力．仮想プリンタに印刷したジョブを一覧から選択し，印刷を実行する．

④印刷データは指定された Ridoc IO Gate 対応プリンタに出力される．

オンデマンド印刷はクライアントから印刷指示を出すときに出力先のプリンタを指定する必要がないため，モバイル PC からの印刷などにも適している．その場合，印刷にはユーザ認証機構がある IPP（Internet Printing Protocol）を利用すればプリンタ設定の煩わしさもなく便利だ．

Ridoc IO Gate には IPP 印刷に対応するためのモジュール RidocIPP がある．以下に Unix をサーバにした場合を例に RidocIPP の動きを説明する（**図 -6**）．

RidocIPP

IPP は HTTP プロトコルを使用してポート番号 631 経由で印刷を行う．

RidocIPP の役割は，（1）IPP オペレーションを受け付けて，（2）その内容を解釈し，（3）Ridoc IO Gate の対応する印刷コマンドを実行し，（4）結果を IPP クライアントに返すことにある．

RidocIPP が起動時に読み込む IPP オペレーションと Ridoc IO Gate の印刷コマンドの対応表は**表 -1** のとおり．

IPP 認証

RidocIPP は IPP 認証に apache の認証機能を利用する．その仕組みは，IPP オペレーションを受け取ると，RidocIPP は apache の認証が必要なページ（以降，「認証ページ」）にアクセスし，IPP クライアントから受け取ったユーザ名とパスワードで認証を試みるというものである．

この認証ページの初期値は http://localhost/ipp/ に設定しているが，Basic 認証の場合を例に，認証ページのディレクトリに作成する .htaccess ファイルと，httpd.conf ファイルの内容を示す．

なお，例における apache のルートディレクトリ（ServerRoot）は /opt/rgate/www/htdocs であり，ユーザ認証用ファイルは /opt/rgate/sys/etc/htpasswd とする．

[認証ページの .htaccess の内容]

```
AuthName        "IPP Authentication"
AuthType        Basic
AuthUserFile    /opt/rgate/sys/etc/htpasswd
AuthGroupFile   /dev/null
 <Limit GET POST>
require valid-user
 </Limit>
```

[httpd.conf の内容]

```
...(中略)...
 <Directory "/opt/rgate/www/htdocs/ipp">
   AllowOverride AuthConfig
</Directory>
 ...(中略)...
```

apache は対応モジュールを組み込めば LDAP や Radius

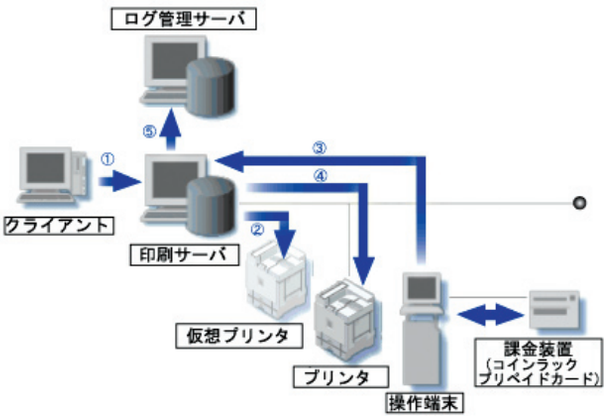

図 -7　課金印刷の流れ

などで認証することも可能だ．この apache の認証機能を利用することで RidocIPP も apache 同等の柔軟な認証手段を提供している．

課金印刷

ユーザは，オンデマンド印刷と同様に操作端末から印刷ジョブを選択して印刷．画面の請求金額をコインラックなど課金装置で入金，精算する（図 -7）．

図中の①～④まではオンデマンドと同様のオペレーションとなるが，最後の⑤で以下に述べるように実印刷枚数に基づく課金の請求がされる．課金情報などは印刷サーバ経由でログ管理サーバに保存される．

実印刷枚数に基づく課金

印刷物の課金金額を決定する要因には，印刷枚数，用紙サイズ，カラー／白黒などがある．それら情報の取得方法は大別して 2 つに分類できる．その 1 つは OS のプリントシステムから取得もしくは直接印刷データを解析する方法．もう 1 つはプリンタから取得する方法だ．前者はソフトウェアだけで手軽に実装できる反面，プリンタで紙づまりなどのトラブルが発生した場合に何枚印刷されたのか不正確になりやすい．それと比べて後者は実印刷枚数をプリンタから取得する方法なのでトラブル発生時も正確であり，この方法を採用したほうが課金システムとしての信頼性は高くなるといえる．

Ridoc IO Gate は実印刷枚数に基づく後者の課金方式を採用しているが，その印刷枚数を取得する方法を示す．

MIB

プリンタの枚数カウンタは MIB（Management Information Base）で管理されており，その値は SNMP で取得することができる．MIB にはベンダ共通（標準）の MIB と各ベンダ独自に拡張した MIB がある．

標準 MIB で分かるのはたとえば次のような情報だ．

- プリンタの累計印刷枚数
- プリンタの状態（エラー，印刷可，印刷中，ウォームアップ中など）
- プリンタの状態がエラーのときその理由（紙なし，トナーなし，カバーオープン，紙づまり，オフライン，サービスコールなど）

残念ながら，標準 MIB では課金に必要なジョブ単位の印刷枚数，用紙サイズ，カラー／白黒といった情報は取得できないため，Ridoc IO Gate ではそれらの情報をリコー独自に拡張した MIB などから取得している．

ジョブ単位の情報を MIB から取得するには，印刷ジョブの先頭に次のような PJL を付加しなければいけない．

```
@PJL SET RIOGTRACKID="1331"
←意味：この印刷ジョブのIDを「1331」に設定
```

以降，RIOGTRACKID の値をキーに MIB を検索することで，このジョブのステータスや印刷枚数などが取得できるようになる．

さらなる TCO 削減に向けて

教育用計算機システムを取り巻く環境はとどまることのないコンピュータ技術の進歩に合わせて常に変化しおり，Ridoc IO Gate が提供する TCO 削減のソリューションもそれに合わせて進化しつづけていく必要がある．

中でも「いつでもどこでも印刷ができる」ユビキタス印刷への対応は急務であり，その第 1 弾として入出力デバイスに USB メモリを利用する課金印刷を近々リリースする．

perl/shell といったスクリプトでカスタマイズが容易な Linux ベースのシステムと，運用管理が容易な Windows ベースのシステム．今後もこの 2 本立ての構成でさまざまな教育用計算機システムの TCO 削減に寄与していくつもりだ．

［おことわり］

本編中のポストスクリプトや PJL は，Ridoc IO Gate で実際に使用しているものとアルゴリズムや考え方は同じであるが，変数名などは意図的に変えている．

参考文献

1）Hastings,T. : Internet Printing Protocol/1.1:Model and Semantics, September 2000, RFC2911.

（平成 16 年 2 月 2 日受付）